

**UNIVERSIDADE NOVA DE LISBOA**

**Faculdade de Ciências e Tecnologia**

**Depº de Engenharia Electrotécnica**

Rede Sem Fios de Microcontroladores com Acesso Remoto Aplicada à Domótica

**Por:**

Gustavo José Henriques Patrício

Dissertação apresentada na Faculdade de Ciências e  
Tecnologia da Universidade Nova de Lisboa para obtenção do  
Grau de Mestre em Engenharia Electrotécnica e de  
Computadores

**Orientador:** Prof. Doutor Luís Filipe dos Santos Gomes

Lisboa

2009

## **Agradecimentos**

Ao Prof. Doutor Luís Gomes por todo o apoio e orientação que tornaram esta dissertação possível, contribuindo também para a minha formação.

Ao Eng. Rogério Piteira por desde sempre me apoiar, motivar e cativar tanto para este trabalho como para toda esta área.

A todo o pessoal da Dynasys que me acolheu, nomeadamente ao Eng. Armando Nunes, Eng. Carlos Dias e D. Zélia. Quero agradecer em especial ao Eng. João Prudêncio cujos ensinamentos, apoio e orientação foram fundamentais para esta dissertação, assim como para toda a minha formação.

A todos os meus colegas e amigos que directa ou indirectamente me ajudaram e apoiaram.

Aos meus Pais, irmãos e familiares que sempre me apoiaram, e ainda à Rita por tudo.

## Sumário

Nesta dissertação apresenta-se o desenvolvimento de um sistema formado por uma rede sem fios de microcontroladores. A rede é capaz de desempenhar funções de monitorização e actuação (sensores-actuadores), quer aplicadas à área da domótica, quer à área das redes de sensores sem fios (Wireless Sensor Networks - WSN). A rede em si é formada por dois tipos de elementos, um coordenador e vários nós. O sistema permite o seu controlo por acesso remoto através de uma página Web. A concretização deste sistema serve ainda como plataforma para a validação da ferramenta de geração automática de código PNML2C. Esta ferramenta parte de um modelo comportamental expresso em Redes de Petri (RdP) IOPT (Input Output Place Transition), e da sua representação em PNML (Petri Net Markup Language), de forma a gerar código C que o implementa. Os nós são implementados recorrendo a esta ferramenta, e os seus resultados analisados. Neste trabalho foi realizada uma rede de sensores-actuadores, capaz de desempenhar funções de monitorização e actuação típicas, recorrendo a componentes de baixo custo, tendo a sua aplicação em tarefas de domótica sido validada através de um protótipo laboratorial. Já a ferramenta PNML2C, embora não concluída, apresentou resultados bastante positivos, deixando bons indícios. A ferramenta foi integrada no ambiente de desenvolvimento e revelou-se capaz de efectuar a tradução fiel de um modelo para a sua implementação.

## Abstract

This dissertation presents the development of a wireless networked microcontroller system. The network can be used for general-purpose monitoring and controlling activities, as well applied to the home automation (Domotics) area and to Wireless Sensor Networks (WSN) area. The network is composed by two elements, a coordinator and the nodes. This system can be controlled remotely through a Web page. The system development framework was also used as a platform to validate the PNML2C tool that is able to generate code automatically. This tool generates C code that implements a model expressed by an IOPT (Input Output Place Transition) Petri Net model represented using the PNML (Petri Net Markup Language) format. The network nodes are implemented using this tool, and its results are analyzed. On this dissertation, a fully functional sensor-actuator network was implemented, amenable for monitoring and actuating activities, relying only on low cost components. A laboratory prototype was developed allowing validation of the results. The PNML2C tool, although not finished, presented positive results. This tool was successfully integrated in the development framework and revealed able to produce an executable translation from the IOPT model amenable for supporting its implementation.

## Simbologia e notações

$\mathbb{N}$  - Conjunto dos números naturais

$\in$  - Pertence ao conjunto

$\sigma$  - Sequência de disparos de uma Rede de Petri

AC – Ar Condicionado

ACKC - Acknowledge Coordinator

ACKN - Acknowledge Node

ACT – Action

AJAX - Asynchronous Javascript And XML

ARP - Address Resolution Protocol

ASCII - American Standard Code for Information Interchange

AW - Asynchronous Wrapper

BOM - Bill Of Materials

CCIPN - Coloured Control Interpreted Petri Nets

CPU – Central Processing Unit

CSMA – Carrier-Sense Multiple Access

DHCP - Dynamic Host Configuration Protocol

DNS – Domain Name System

EEPROM - Electrically Erasable Programmable Read-Only Memory

EHS - European Home Systems Protocol

EI – Edifício Inteligente

EIB - European Installation Bus

ETSI - European Telecommunications Standards Institute

EUA – Estados Unidos da América

FCC - Federal Communication Commission

FFD – Full Function Device

GALS - Globally Asynchronous Locally Synchronous

GPS – Global Positioning System

HDL - Hardware Description Language

HTML - HyperText Markup Language

HTTP - Hypertext Transfer Protocol

IC - Integrated Circuit

ICD – In-Circuit Debug  
ICMP - Internet Control Message Protocol  
IDE - Integrated Development Environment  
IOPT - Input-Output Place-Transition  
IP - Internet Protocol  
IrDA - Infrared Data Association  
ISM - Industrial, Scientific and Medical radio bands  
LAN – Local Area Network  
LCD – Liquid Crystal Display  
LED - Light-Emitting Diode  
LS - Localmente Síncrono  
MAC – Medium Access Control  
MCU - Microcontroller Unit  
MdC – Modelos de Computação  
MEMS - Microelectromechanical Systems  
MOF - Meta-Object Facility  
MPFS – Microchip File System  
NES - Networked Embedded Systems  
NEST - Networked Embedded System Technology  
OS - Operating System  
OSI - Open Systems Interconnection  
PC – Personal Computer  
PCB - Printed Circuit Board  
PCC - Pausable Clock Control  
PDA – Personal Digital Assistant  
PDF - Portable Document Format  
PHY – Physical Layer  
PLC - Power Line Communication  
PNML - Petri Net Markup Language  
PPS - Peripheral Pin Select  
PSV - Program Space Visibility  
QVGA - Quarter Video Graphics Array  
RADAR - Radio Detection And Ranging

RAM - Random-Access Memory  
RdP – Rede de Petri  
RdP-R – Redes de Petri Reactivas  
RdP-RH - Redes de Petri Reactivas e Hierarquicas  
REQ - Request  
RF – Rádio Frequência  
RFD – Reduced Function Device  
RFID - Radio-Frequency Identification  
RTCC – Real Time Clock Calendar  
SoC – System on Chip  
SPI - Serial Peripheral Interface  
SSL- Secure Sockets Layer  
SVG - Scalable Vectorial Graphics  
TCP - Transmission Control Protocol  
TDMA - Time-Division Multiple Access  
UDP - User Datagram Protocol  
UML – Unified Modeling Language  
USB - Universal Serial Bus  
VHDL - VHSIC Hardware Description Language  
VHSIC - Very-High-Speed Integrated Circuits  
VLSI - Very-Large-Scale Integration  
WLAN – Wireless Local Area Network  
WSN - Wireless Sensor Network  
WPAN - Wireless Personal Area Network  
XMI - XML Metadata Interchange  
XML - Extensible Markup Language

## Índice de Matérias

Agradecimentos	1
Sumário	2
Abstract	3
Simbologia e notações	4
Índice de Matérias	7
Índice de Figuras	10
Índice de Quadros	13
Introdução	14
Objectivos	14
Motivação	15
Estrutura da Dissertação	17
1- Domótica	19
1.1- Introdução	19
1.2- Conceitos e Arquitecturas	23
1.3- Tecnologias	25
1.3.1- Protocolo KNX	28
1.3.2- X10	29
1.3.3- INSTEON	30
1.3.4- ZigBee	31
1.3.5- Outras Tecnologias	34
1.4- Domótica em Portugal	34
2- Sistemas Embutidos em Rede	38
2.1- Introdução	39
2.2- Hardware	40
2.3- Software	42
2.4- Aplicações	45
3- Formalismos de Modelação	48
3.1- Fluxogramas	50
3.2- Redes de Petri	53
3.2.1- Enquadramento Histórico das Redes de Petri	53
3.2.2- Conceito de Redes de Petri	54
3.2.3- Estrutura das Redes de Petri	55
3.2.4- Disparo das transições	58



3.2.5- Modelação com Rede de Petri	60
3.2.6- Classes de Redes de Petri	63
3.2.7- PNML	64
3.3- UML	66
3.3.1- Diagrama de Casos de Uso	67
3.3.2- Diagramas de Sequência	69
4- Sistemas GALS	70
4.1- Introdução	70
4.2- Conceitos dos Sistemas GALS	71
4.2.1- Encapsulamento Assíncrono	72
4.2.2- Controlador de Portas	73
4.2.3- Gerador do sinal de relógio	75
4.3- Resultados dos sistemas GALS	75
4.4- Vantagens e desvantagens	78
4.5- Utilizações de Sistemas GALS	80
4.6- Enquadramento	81
5- Ferramentas Utilizadas	83
5.1- Ferramentas para as Redes de Petri	83
5.1.1- Snoopy-IOPT	83
5.1.2- Conversor PNML2C	83
5.2- Ferramentas de desenvolvimento	85
5.2.1- MPLAB e Compilador	85
5.2.2- Stack TCP-IP	86
5.3- Ferramentas de documentação	87
5.3.1- Doxygen	87
5.4- Ferramentas de Debug	87
5.4.1- Wireshark	87
5.4.2- ZENA Network Analyzer	88
5.5- Hardware Utilizado	89
5.5.1- PIC24FJ128GA010	89
5.5.2- Placa Explorer16	89
5.5.3- Kit PICDEM Z	90
5.5.4- Ethernet PICtail Plus	91
5.5.5- PICtail Plus 2.4GHz RF	92
5.5.6- Comando IrDA – KBC56A	92

5.5.7- ICD2	93
6- Contribuições	94
6.1- Descrição do Sistema	94
6.2- Casos de uso	96
6.3- Arquitectura do Sistema	98
6.3.1- Coordenador	99
6.3.2- Protocolo	112
6.3.3- Nó	116
6.4- Teste e Resultados	122
6.4.1- Testes Unitários	122
6.4.2- Testes de Sistema	125
6.4.3- Resultados da ferramenta PNML2C	128
6.4.4- Modelo de Aplicação	129
7- Conclusões	133
Referências	136
Anexo I – Manual de utilização	145

## Índice de Figuras

Figura 1.1 - Exemplo de um sistema domótico, reproduzido de [HAI, 09]	22
Figura 1.2 – Protocolos sem fios, adaptado de [Masters, 08]	26
Figura 1.3 – Rede em Estrela [Masters, 08]	26
Figura 1.4 - Rede em Árvore [Masters, 08]	27
Figura 1.5 - Rede em Malha [Masters, 08]	27
Figura 1.6 – Lâmpada X10, extraído de [EuroX10, 09]	29
Figura 1.7 - Módulo X10, extraído de [EuroX10, 09]	29
Figura 1.8 – Conversor X10, extraído de [EuroX10, 09]	30
Figura 1.9 – Comando X10, extraído de [EuroX10, 09]	30
Figura 1.10 – Pilha de camadas ZigBee, adaptado de [Masters 08]	31
Figura 2.1 – Topologia típica WSN	39
Figura 2.2 – Esquema de um <i>mote</i>	41
Figura 3.1 - Terminal	50
Figura 3.2 - Processamento	51
Figura 3.3 - Decisão	51
Figura 3.4 – Entrada Manual	51
Figura 3.5 - Exibição	51
Figura 3.6 - Documento	51
Figura 3.7 - Dados	51
Figura 3.8 – Dados Gravados	51
Figura 3.9 – Preparação	51
Figura 3.10 – Processo Predefinido	52
Figura 3.11 – Operação Manual	52
Figura 3.12 - Conector	52
Figura 3.13 - Linha	52
Figura 3.14 – Exemplo de Fluxograma	52
Figura 3.15 - Lugar	55
Figura 3.16 - Transição	56
Figura 3.17 - Arco	56
Figura 3.18 – Exemplo de uma Rede de Petri, adaptado de [Murata, 89]	57
Figura 3.19 – Rede de Petri marcada, extraído de [Reis, 08]	58
Figura 3.22 – Disparo de uma transição (arcos com pesos).	59
Figura 3.20 – Transição desabilitada	59
Figura 3.21 – Disparo de uma transição. a) situação inicial. b) situação após o disparo	59
Figura 3.23 – Paralelismo	60
Figura 3.24 - Conflito	61
Figura 3.25 - Confusão	61
Figura 3.26 - Sincronização	61
Figura 3.27 – Gestão de recursos	61
Figura 3.28 – Lugares complementares	62
Figura 3.29 - Memória	62
Figura 3.30 – Arco de teste	62
Figura 3.31 – Ferramentas Fordesign, adaptado de [Fordesign, 07]	66

Figura 3.32 – Exemplo de diagrama de casos de uso, adaptado de [Rumbaugh et al, 99]	68
Figura 3.33 - Exemplo de diagrama de sequência, adaptado de [Rumbaugh et al, 99]	69
Figura 4.1 Comunicação assíncrona entre módulos, adaptado de [Muttersbach et al, 00]	70
Figura 4.2 - Módulo funcional num sistema GALS, adaptado de [Muttersbach et al, 00]	73
Figura 4.3 - Esquema de uma Porta tipo-D [Muttersbach et al, 00]	74
Figura 4.4 - Esquemático de um controlador D-type [Muttersbach et al, 00]	74
Figura 4.5 - Processador síncrono (base), extraído de [Iyer-Marculescu, 02]	76
Figura 4.6 - Processador GALS, extraído de [Iyer-Marculescu, 02]	76
Figura 4.7 - Desempenho do processador GALS relativo ao de base, de [Iyer-Marculescu, 02]	77
Figura 4.8 - Média de <i>slip</i> , extraído de	77
Figura 4.9 - Consumo do processador GALS relativo ao de base, extraído de [Iyer-Marculescu, 02]	78
Figura 5.1- Snoopy-IOPT	83
Figura 5.2 – Arquitetura do código gerado, extraído de [Pais, 04]	84
Figura 5.3 – MPLAB IDE, extraído de [Microchip, 09]	85
Figura 5.4 – Stack TCP-IP da Microchip, extraído de [Microchip, 09]	86
Figura 5.5 – Captura no Wireshark	88
Figura 5.6 – Placa Zena Network Analyzer, extraído de [Microchip, 09]	88
Figura 5.7 – Interface gráfico do ZENA Network Analyzer, extraído de [Microchip, 09]	88
Figura 5.8 – Diagrama de blocos PIC24F, extraído de [Microchip, 09]	89
Figura 5.9 – PIC32 PIM, extraído de [Microchip, 09]	90
Figura 5.10 – Explorer 16, extraído de [Microchip, 09]	90
Figura 5.11 – Kit PICDEM Z, extraído de [Microchip, 09]	91
Figura 5.12 – Ethernet PICTail Plus, extraído de [Microchip, 09]	91
Figura 5.13 - PICTail Plus 2.4GHz RF, extraído de [Microchip, 09]	92
Figura 5.16 – Microchip ICD2, extraído de [Microchip, 09]	93
Figura 5.14 - Receptor KBC56A, extraído de [Noritake, 09]	93
Figura 5.15 – Comando KBC56A, extraído de [Noritake, 09]	93
Figura 6.1 – Topologia da rede	95
Figura 6.2 – Diagrama de sequência do Protocolo	96
Figura 6.3 – Casos de uso do Sistema	96
Figura 6.4 – Casos de uso do coordenador	97
Figura 6.5 – Casos de uso dos nós	98
Figura 6.6 – <i>Hardware</i> do Coordenador	100
Figura 6.7 – Diagrama de blocos do coordenador	101
Figura 6.8 – Fluxograma do coordenador	111
Figura 6.9 – Mensagem HELLO	113
Figura 6.10 – Mensagem ACKC	113
Figura 6.11 – Mensagem REQ	113
Figura 6.12 – Mensagem ACT	114
Figura 6.13 – Mensagem ACKN	114
Figura 6.14 – <i>Hardware</i> do nó	117
Figura 6.15 – Diagrama de blocos do nó	117
Figura 6.16 – Fluxograma do nó	119
Figura 6.17 – RdP do nó	120
Figura 6.18 – 1º Teste unitário	123
Figura 6.19 – 2º Teste unitário	124

Figura 6.20 – 1º Teste de sistema	126
Figura 6.21 – 2º Teste de sistema	127
Figura 6.22 – Planta do modelo de aplicação	130
Figura 6.23 – Esquema 3D do modelo de aplicação	131
Figura 6.24 – Protótipo do modelo de aplicação	131
Figura 6.25 – Segundo Protótipo	132
Figura I.0.1 – Acesso à configuração	145
Figura I.0.2 – Secção dos nós	146
Figura I.0.3 – Secção das tarefas agendadas	147
Figura I.0.4 – Secção das regras	148
Figura I.0.5 – Secção do estado dos sensores	149

## Índice de Quadros

Tabela 3.1 - Classes das RdP [Lino, 03]	63
Tabela 6.1 – Formato do tempo no RTCC, adaptado de [PIC24F, 07]	102
Tabela 6.2 – Máscaras de Alarme do RTCC, adaptado de [PIC24F, 07]	103
Tabela 7.1 – Custo das ferramentas de desenvolvimento	133
Tabela 7.2 – Custo dos componentes	134

## Introdução

### Objectivos

Esta dissertação centra-se em dois objectivos principais. O primeiro objectivo é o projecto, desenvolvimento e implementação de um sistema que forme uma rede de sensores-actuadores. O segundo objectivo consiste na validação da ferramenta de geração automática de código PNML2C.

A rede de sensores deverá ser capaz de recolher informação dos seus vários elementos e reagir conforme situações específicas. Este sistema deverá desempenhar funções quer no âmbito da domótica, quer no âmbito das redes de sensores sem fios (Wireless Sensor Networks - WSN).

Nesta rede existirá um elemento designado coordenador, onde residirá toda a inteligência do sistema. Cada elemento representa um subsistema, que quando associado a outros formam uma rede. Além deste existirão os nós que irão recolher informação dos sensores, enviando-a ao coordenador e accionar se necessário os seus actuadores. A comunicação entre estes elementos deverá ser efectuada sem fios por rádio frequência (RF).

O sistema terá que ser autónomo na medida em que deverá saber como reagir a situações previamente definidas, sem que para isso o utilizador tenha que interferir. Deverá ainda desempenhar tarefas que o utilizador agende para um momento específico.

De modo a configurar e visualizar o estado do sistema haverá um interface com o utilizador. Este será implementado no coordenador e será composto por uma página *Web*. Assim será possível ao utilizador do sistema a interacção com o mesmo remotamente.

Com a concretização deste primeiro objectivo, o sistema desenvolvido servirá como plataforma para a realização do segundo objectivo. Neste a modelação dos nós da rede deverá ser efectuada através de Redes de Petri (RdP). Partindo desta RdP será gerado código automaticamente que será implementado e testado. Assim este segundo objectivo centra-se na validação da ferramenta académica que permite a geração de código C, partindo de um modelo expresso por RdP. Esta plataforma irá pôr a interagir

um sistema composto por vários elementos cujo desenvolvimento segue diferentes metodologias.

## **Motivação**

A realização deste trabalho foi motivada por diversos factores. Em primeiro lugar pelo gosto pessoal em desenvolver pequenos equipamentos controlados por microcontroladores (Microcontroller Unit - MCU) com funcionalidades inovadoras. O desenvolvimento deste tipo de sistemas é sem dúvida, por agora, o que mais gosto e realização confere ao autor desta dissertação.

Desde sempre que houve um fascínio pelos sistemas embutidos, e a actual capacidade de lhes adicionar funcionalidades avançadas torna-os ainda mais apelativos.

Este trabalho reúne várias componentes interessantes. Uma dessas componentes é o desenvolvimento de um sistema embutido através da programação de microcontroladores. Outra é a possibilidade de conferir ao sistema um interface remoto através de uma página *Web*. Já a comunicação por rádio frequência é sem dúvida uma componente que confere aos sistemas uma versatilidade única.

Este gosto pelos sistemas embutidos desde sempre existiu. Produzir um pequeno programa que faça acender um LED (Light-Emitting Diode) sempre cativou muito mais o autor desta dissertação do que o tradicional HelloWorld da programação para aplicações PC (Personal Computer). Actualmente é possível em vez de acender um LED efectuar algo tão avançado como modificar uma página *Web*, enviar um e-mail ou mesmo representar esse “LED” aceso num ecrã sensível ao toque (*touch-screen*). Antigamente o interface era apenas formado por componentes como botões, LEDs e potenciómetros, sendo muito pouco apelativo para o utilizador. Neste momento é possível criar interfaces tão ricos como os criados em aplicações típicas para PC, ou ainda interfaces desenhados para a *Web* que são e continuarão a ser, sem dúvida, os que os utilizadores mais gostam e acostumados estão.



A possibilidade de visitar um equipamento remotamente através da Internet é uma ferramenta extremamente poderosa e com inúmeras potencialidades. Tanto a telemetria como o controlo remoto são áreas muito interessantes. A inclusão desta funcionalidade ao trabalho desenvolvido foi sem dúvida motivadora.

Já a comunicação sem fios por RF é também uma característica cativante. Actualmente pode-se verificar que as tecnologias sem fios têm tido uma proliferação e aceitação muito grande. Aplicado aos sistemas embutidos, vemos e aguardamos o papel que, por exemplo, o RFID poderá vir a ter na nossa sociedade.

De certo modo a motivação para esta dissertação surge com os recentes e grandes avanços na área dos sistemas embutidos, que aliados aos interesses pessoais, constituem um factor impulsionador ao desenvolvimento deste trabalho.

Mas não são apenas estes que motivam esta dissertação. A possibilidade de modelar partes do sistema através de RdP e gerar o código que a implementa, é algo extremamente aliciante. Por vezes, certos formalismos e modelações parecem não ter equivalência/aplicabilidade em sistemas reais. Essa falha na concretização de certos modelos pode ser desmotivante. Neste caso é extremamente motivador o uso de um formalismo de modelação que implemente na realidade o modelo descrito. Ferramentas deste tipo podem ser um factor decisivo no uso e aplicação de formalismos deste género.

A utilização de todos estes “ingredientes” para a implementação de um sistema aplicado à área da domótica ou ainda das WSN é muito motivador. Isto porque são áreas que estão em franca expansão e em que existem expectativas elevadas na sua utilização, sendo um desafio introduzir atitudes associadas ao desenvolvimento baseado em modelos, em particular baseado em RdP, no seu desenvolvimento.

## Estrutura da Dissertação

Esta dissertação encontra-se dividida em sete capítulos. Começa-se, no primeiro, por introduzir a área da domótica. Inicialmente será feita uma descrição desta área e explicados alguns conceitos relacionados. Serão apresentadas várias aplicações onde a domótica se torna relevante. Serão abordadas algumas das tecnologias que dão suporte a este conceito e ainda será efectuada uma caracterização do estado actual da domótica em Portugal.

No segundo capítulo serão abordadas algumas questões relacionadas com as redes de sistemas embutidos (Networked Embedded Systems - NES). Dentro desta vasta área serão abordados os temas que dizem respeito mais especificamente às WSN. Irão ser referidas algumas das questões relacionadas com estas redes, como as implementar e quais as grandes dificuldades e desafios que esta área enfrenta. No fim são apresentadas várias aplicações destas redes.

No terceiro capítulo surge uma descrição de alguns formalismos de modelação. Será apresentada a motivação para o uso de formalismos deste género e em seguida serão abordados três formalismos de modelação. Os fluxogramas são descritos e exemplificados. As RdP serão descritas em detalhe, apresentado uma caracterização e definição das mesmas. Já o UML será brevemente abordado, sendo apenas descritas duas das muitas técnicas de modelação que este disponibiliza.

O quarto capítulo apresenta uma caracterização de um conceito para concepção de sistemas complexos, os sistemas GALS (Globally Asynchronous Locally Synchronous). Nesta caracterização explica-se o conceito, assim como as suas implementações. Em seguida são enumeradas as suas vantagens e desvantagens e são ainda apresentados os resultados de vários testes de desempenho. No final é feito um enquadramento destes sistemas nos temas desta dissertação.

O quinto capítulo apresenta todas as ferramentas que foram utilizadas neste trabalho. São referidas as ferramentas, quer de *hardware* quer de *software* que contribuíram e constituíram o trabalho desenvolvido.

É no sexto capítulo que será apresentado todo o trabalho elaborado. Nesta apresentação muitos aspectos serão abordados, desde a identificação dos objectivos/requisitos à concretização do sistema, passando pelas arquitecturas e modelos implementados.

Neste capítulo discute-se as opções tomadas ao longo do trabalho, assim como as especificidades inerentes do sistema. Na parte final deste capítulo irão ser apresentados alguns dos testes e resultados conseguidos de forma a sustentar o capítulo seguinte.

No sétimo capítulo será feito todo o balanço deste trabalho. Serão interpretados os resultados e apresentadas as conclusões finais deste trabalho assim como perspectivas futuras.

## **1- Domótica**

Neste capítulo vai-se descrever sumariamente o que é Domótica, em que enquadramento surge, e ainda que tecnologias dão suporte a este conceito.

### **1.1- Introdução**

O natural instinto de protecção obrigou, desde sempre, o homem a procurar abrigo. Começou por habitar em cavernas, depois em cabanas e posteriormente, com os avanços da civilização, em edifícios. À medida que estes foram evoluindo, surgiu a necessidade de controlar os seus espaços de modo a adaptá-los a diversas condicionantes, nomeadamente as alterações meteorológicas. Assim, para controlar os fluxos de ar surgiram as portas e as janelas; para controlar a luminosidade no interior, bem como por questões de privacidade, surgiram as portadas e os estores. Mais recentemente, com o aparecimento da electricidade, surgiram os interruptores para controlar o funcionamento dos dispositivos eléctricos. Todos estes “controles” têm uma particularidade em comum: a necessidade de acção humana directa.

Com a evolução tecnológica foi possível adicionar alguns automatismos através do uso de elementos básicos como relés ou termóstatos. Estes automatismos, que surgiram nos anos 60, eram designados de automatização local, uma vez estes apenas reagiam a certos estímulos locais e não passam a informação. Já nos anos 70 com a evolução electrónica e dos sistemas baseados em microprocessadores a automação dos edifícios converge para sistemas centralizados em que a informação era concentrada numa central.

É então nos anos 80, motivados pelos novos requisitos de conforto e segurança, que surgiram sistemas para detecção de incêndios, controlo de iluminação climatização, entre outros [Nunes, 04a].

Foi com o intuito de automatizar e controlar estes processos que surgiram os Edifícios Inteligentes (EIs). Inicialmente os EIs foram criados principalmente com o objectivo de redução dos custos energéticos assim como dos de utilização e manutenção dos edifícios.

Este conceito tem sido interpretado de diversas formas, distorcido e redefinido. Várias definições são referidas em [Gomes, 97], das quais se apresenta a seguinte. Um “Edifício Inteligente deve fornecer um ambiente confortável e adaptável para o utilizador, flexível e com custos eficazes para a organização e atractivo e capaz para o proprietário; cada sistema irá interagir, de alguma forma, com todos os outros e todos os sistemas devem trabalhar em harmonia se pretender alcançar benefícios plenos para o utilizador” [Shemie, 97].

A concepção de um EI é uma área de aplicação multidisciplinar, implicando diferentes domínios do conhecimento a interagir no sentido benéfico não só dos ocupantes ou utilizadores destes edifícios, mas também dos proprietários dos mesmos. Estes benefícios são difíceis de quantificar, quer em termos económicos quer em termos ecológicos, mas podem ser associados à poupança de energia. Dos domínios do conhecimento envolvidos com contribuições relevantes nesta área destacam-se a arquitectura, as engenharias civil, electrónica e de computadores, nomeadamente de sistemas, controlo e telecomunicações [Gomes, 97].

A Domótica (em inglês mais conhecida como *home automation*, *smart home* ou mesmo *domotics*) surge como um ramo dos Edifícios Inteligentes mas com aplicação nas habitações. A palavra Domótica é o resultado da junção da palavra latina *Domus*, que significa Casa, com a palavra Robótica. Ao contrário do contexto habitual dos EIs (indústria e militar) a domótica procura fornecer conforto e conveniência aos lares domésticos. A ideia base é automatizar tarefas e rotinas de uma casa e ainda possibilitar a capacidade de interagir com a casa remotamente. Dos domínios do conhecimento envolvidos com contribuições relevantes nesta área destacam-se a arquitectura, as engenharias civil, electrónica e de computadores, nomeadamente de sistemas, controlo e telecomunicações [Gomes, 97]. De entre os objectivos idealizados para uma casa inteligente referem-se os seguintes:

- Controlo de iluminação. Controlando cada lâmpada da casa, permitindo um ajuste energético preciso. Por exemplo, ao configurar-se cada lâmpada segundo um horário de actividade permitido, elimina-se a possibilidade de estas ficarem acesas

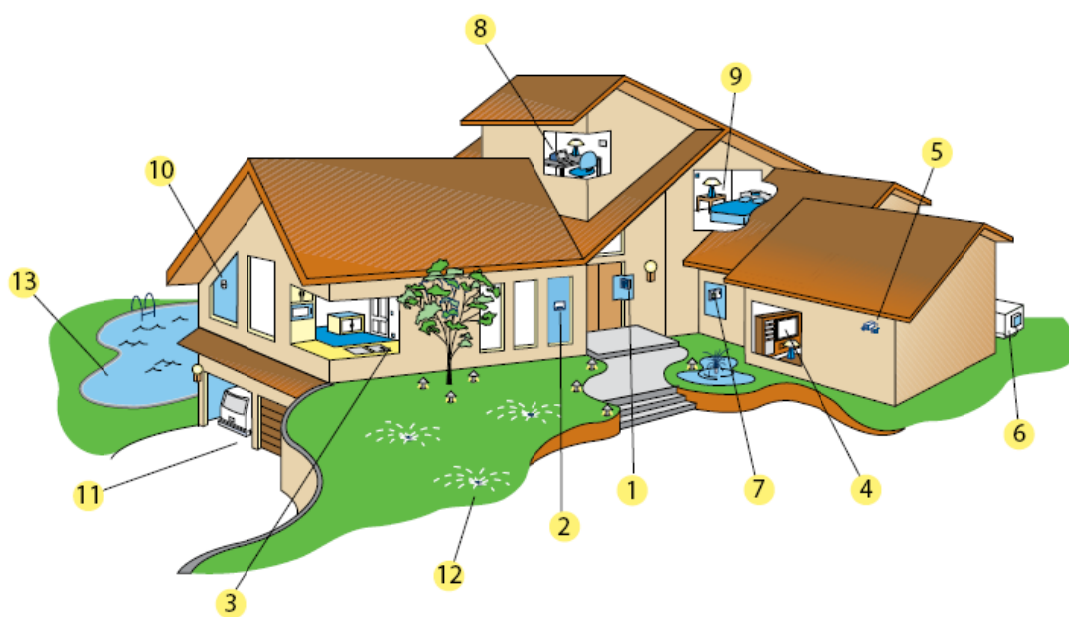
noites inteiras por esquecimento. Ou ainda, através da instalação de sensores, permitir que a lâmpada esteja acesa apenas durante a presença de uma pessoa.

- Limpeza automática da casa. Configurando o horário de funcionamento do sistema de aspiração central, por exemplo, durante a tarde quando ninguém está em casa, evita o incómodo do ruído e possibilita que a casa se encontre limpa quando as pessoas regressam.
- Confeção de refeições. Programando o horário de funcionamento do forno, permite que a comida, previamente preparada, esteja pronta à hora de regresso das pessoas a casa.
- Irrigação de plantas. Configurando o sistema de rega, permite que em períodos pré-definidos seja fornecida a quantidade exacta de água às plantas. Este sistema pode evitar esquecimentos ou ainda ser muito útil em períodos de férias.
- Alimentação automática de animais domésticos. Utilizando um dispositivo que a horas pré-definidas liberta uma porção de ração ao animal.
- Segurança. Instalando sensores em pontos estratégicos permite accionar um sistema de alarme perante a presença de intrusos, ou efectuar o controlo de acessos de forma a gerir o tráfego de pessoas, nomeadamente em pensões ou residências.

As funcionalidades descritas, entre as muitas existentes, são meramente elucidativas, podendo ainda ser criado qualquer tipo de funcionalidade conforme os requisitos pretendidos. A grande vantagem deste tipo de soluções é que toda a gestão da casa está centralizada numa unidade que gere todos estes utensílios/dispositivos conforme os perfis/comportamentos que o utilizador pretende. Dos exemplos anteriormente mencionados foi sempre utilizado o automatismo programado. No entanto existem outras possibilidades, como por exemplo, se a central do sistema estiver com acesso à Internet, o utilizador pode accionar qualquer mecanismo quando quiser ou consultar o

estado de todos os sensores espalhados pela casa. Esta interação remota em tempo-real é uma funcionalidade extremamente importante, pois confere uma poderosa ferramenta aos seus utilizadores.

Por exemplo, se as pessoas da casa estão ausentes no trabalho ou de férias e se apercebem de que o mau tempo vai atingir a área da residência, através desta funcionalidade podem accionar remotamente os mecanismos que baixam os estores, de modo a proteger melhor a casa da intempérie.



**Figura 1.1** - Exemplo de um sistema domótico, reproduzido de [HAI, 09]

Apresenta-se na Figura 1.1 um exemplo de uma casa “domotizada”, em que os vários pontos marcados têm o seguinte significado:

- 1- Consola de controlo de todo o sistema. Permite a gestão da segurança, climatização, iluminação, irrigação, multimédia, etc.
- 2- Sensores para a segurança da casa.
- 3- Iluminação com modos predefinidos, por exemplo, para ver um filme, ou para tomar a refeição.
- 4- Sistema multimédia que pode ser partilhado para todas as divisões.
- 5- Câmaras de vigilância, que podem ser visionadas em qualquer altura, em qualquer divisão.

- 6- Gestão de energia.
- 7- Termostato para a climatização.
- 8- Ligação do sistema à Internet de modo a que o sistema esteja acessível em qualquer parte do mundo.
- 9- Ligação do sistema à rede telefónica de modo a que seja também acessível pela linha telefónica.
- 10- Sensor de movimentos para a detecção de intrusos.
- 11- Detecção de veículos, de modo a que com a aproximação de alguém as luzes se acendam e surjam no ecrã as imagens da câmara no exterior.
- 12- Sistema de irrigação gerido pelo sistema. Com programas estabelecidos que podem ser interrompidos em dias de chuva.
- 13- Gestão dos ciclos da piscina para a manutenção da qualidade da água.

## 1.2- Conceitos e Architecturas

Um sistema domótico é composto por vários nós que, espalhados pelo edifício, formam uma rede. Estes nós, instalados em locais específicos conforme as suas funções, podem ser agrupados de acordo com a seguinte classificação:

- **Sensores** - nós que apenas recolhem informação e a enviam para outro nó. Por exemplo um nó que apenas recolhe informação sobre a temperatura e humidade de uma divisão.
- **Actuadores** - nós que têm a capacidade de actuar sobre algo. Um exemplo será um nó num interruptor de uma lâmpada, que a acende ou apaga.
- **Sensores-actuadores** - nós que tanto têm a capacidade de obter informação do mundo exterior, como de actuar sobre algum dispositivo. Considerando os dois exemplos anteriores, em vez de ter dois nós na mesma divisão, ter-se-ia apenas um com a capacidade de ligar e desligar a lâmpada, mas ainda estaria equipado com os sensores de temperatura e de humidade.



Em termos de inteligência do sistema este pode ter as seguintes arquitecturas.

- **Centralizada** - quando há um nó, que costuma ser designado como coordenador, recebe toda a informação dos sensores e de qualquer outro interface com o utilizador, processa essa informação e reage enviando ordens aos actuadores, conforme os comportamentos pré-definidos. No trabalho desenvolvido nesta dissertação foi esta a arquitectura utilizada. Esta topologia pode ser vantajosa na economia de recursos, pois os nós da rede, excepto o coordenador, podem ser extremamente simples do ponto de vista de *software* e *hardware*, devido ao facto de não necessitarem de elaborar grandes processamentos. No caso dos sensores devem apenas saber recolher informação e transmiti-la. Já nos actuadores, apenas aguardam uma ordem, e actuam nos dispositivos a que estão associados. Outra vantagem do facto de ter toda a inteligência estar num único nó é que apenas é necessário programar/configurar um nó quando queremos alterar o comportamento do sistema a uma situação. Já no caso de actualização em termos de *firmware* do sistema, apenas é necessário actualizar o coordenador, pois os sensores e actuadores continuaram a desempenhar o mesmo papel básico. Como é óbvio esta arquitectura tem a sua grande desvantagem no caso de o coordenador ficar fora de serviço, o que representa uma falha total no sistema.
- **Distribuída** - quando todos os nós presentes na rede têm a capacidade de processar a informação e reagir. Esta abordagem tem a vantagem de quando algum nó deixa de funcionar, a rede apenas perde as funcionalidades associadas a esse nó e tudo o resto decorre sem problemas.
- **Mista** - quando existem nós capazes de adquirir informação dos sensores, processá-la, reagir e depois ainda a enviam para os outros nós.

### 1.3- Tecnologias

Actualmente existem inúmeras soluções para implementar um sistema domótico. Algumas soluções são específicas de certos fabricantes enquanto outras são normas que entidades definiram de modo a que todos os fabricantes que queiram desenvolver produtos o possam fazer desde que cumpram os requisitos (e muitas vezes paguem algumas taxas e licenciamento).

Existem diversos meios de comunicação que podem ser utilizados nas variadas soluções. Enquanto umas utilizam um só meio, outras utilizam vários, quer por questões de robustez quer por compatibilidade com outros sistemas. Os meios de comunicação utilizados podem ser: rede eléctrica, rádio frequência (RF), Bluetooth, infravermelhos (Infrared Data Association - IrDA), Ethernet formando uma rede de área local (*Local Area Network* - LAN), rede de área local sem fios (*Wireless Local Area Network* - WLAN), cabo coaxial, ou qualquer ligação através de fios tipicamente cabo com fios entrançados (*twisted-pair*).

Como o meio de comunicação utilizado no trabalho desenvolvido nesta dissertação é a RF, em seguida apresenta-se uma caracterização deste meio.

O meio de comunicação RF é muito utilizado devido à sua fácil instalação e nomeadamente ao facto de não necessitar de fios. Neste meio existem regras específicas quanto às frequências utilizadas (bandas ISM) e potências de sinal as quais são estipuladas por entidades reguladoras: a *Federal Communication Commission* (FCC) nos Estados Unidos da América (EUA), a *European Telecommunications Standards Institute* (ETSI) na Europa e a *Industry Canada* no Canadá, entre outras.

Na Figura 1.2 apresenta-se um gráfico comparativo de várias tecnologias sem fios (*wireless*). Nele pode-se verificar que existem muitas especificações sem fios, e que podem ter alcances e taxas de transferência bem diferentes. O meio de comunicação RF encontra-se representado na figura através da elipse que contém o IEEE 802.15.4 que é uma norma para a realização de Redes de área pessoal sem fios (*Wireless Personal Area Networks* - WPAN) de baixo ritmo. Essa elipse contém ainda alguns protocolos mais específicos que serão abordados mais à frente.

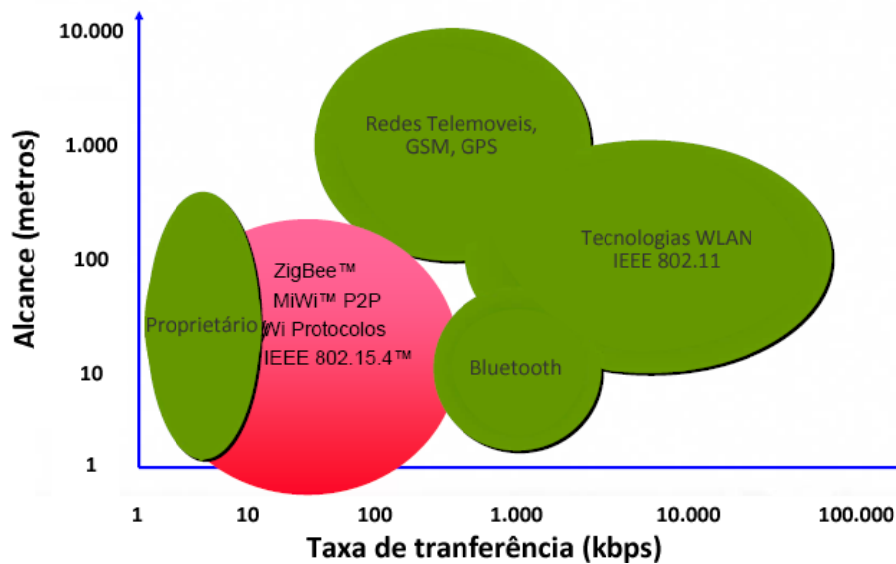


Figura 1.2 – Protocolos sem fios, adaptado de [Masters, 08]

Estas redes RF podem ter as seguintes topologias:

- em estrela;
- em árvore;
- em malha.

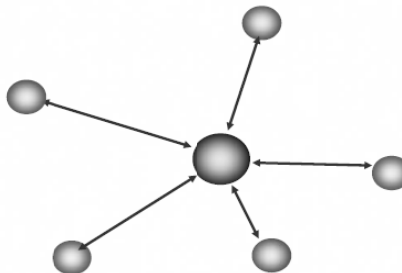
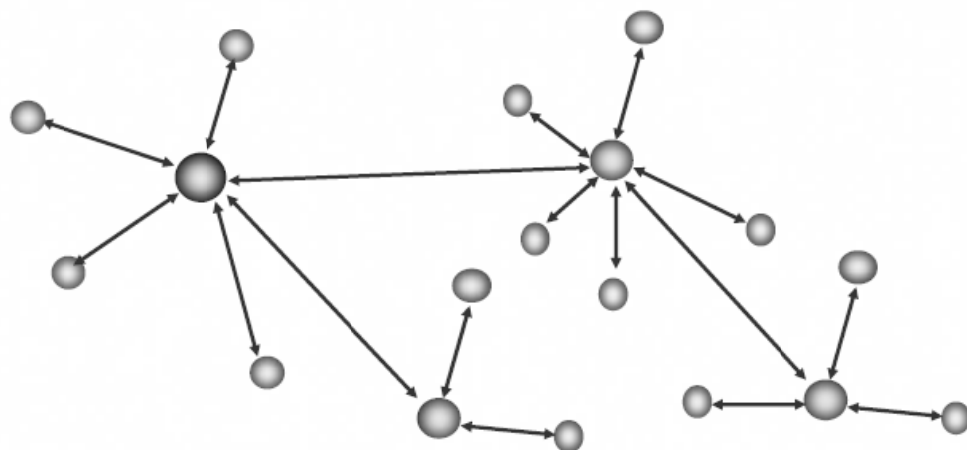


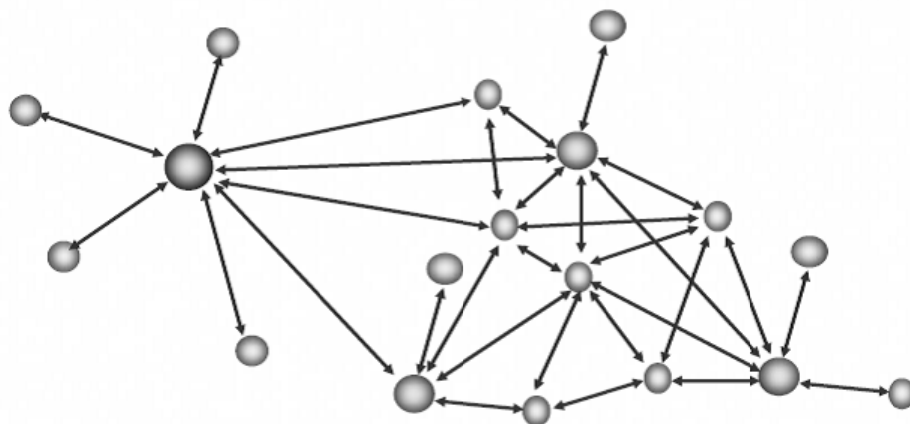
Figura 1.3 – Rede em Estrela [Masters, 08]

Na Figura 1.3 apresenta-se a topologia de rede do tipo estrela. Nesta topologia todos os nós comunicam com o coordenador de rede directamente. A grande desvantagem é que todos os nós da rede têm que estar ao alcance do coordenador, o que em certas soluções não é praticável devido a barreiras arquitectónicas, ou mesmo distâncias longas.



**Figura 1.4** - Rede em Árvore [Masters, 08]

A rede em árvore (*cluster tree*), é apresentada na Figura 1.4. Nesta topologia existem nós com a capacidade de reencaminhar as mensagens de modo a que a mensagem salte de nó em nó até ao seu destino. Esta topologia já requer um maior processamento por parte dos nós que reencaminham, porque necessitam executar o encaminhamento. Nesta topologia já é possível obter qualquer distância, desde que sejam colocados nós capazes de reencaminhar, dentro do alcance dos rádios adjacentes, e o número de saltos (*hops*) não seja superior ao limite do protocolo.



**Figura 1.5** - Rede em Malha [Masters, 08]

A rede em malha (*mesh*), é apresentada na Figura 1.5. Nesta topologia todos os nós têm a capacidade de reencaminhar mensagens excepto os de funções reduzidas. No entanto, existem outras concretizações de rede em malha em que todos os nós repetem

as mensagens. Essas redes são designadas de malha dupla (*dual-mesh*) e um exemplo destas é a INSTEON. Assim, quando existe uma mensagem a enviar na rede, esta pode seguir por vários caminhos até chegar ao destino. Esta topologia é a que apresenta uma maior fiabilidade mas também é a que exige mais recursos de *hardware* e *software*. Isto porque todos os nós são capazes de reencaminhar, logo estão constantemente a retransmitir, o que em termos de gestão de baterias pode ser muito prejudicial.

Em seguida apresentam-se algumas das soluções para sistemas domóticos.

### 1.3.1- Protocolo KNX

É uma norma aberta para o controlo de lares e edifícios. Foi criado em 1999 e é o resultado da fusão entre três normas anteriores, o European Home Systems Protocol (EHS), o BatiBUS, e o European Installation Bus (EIB). Actualmente a entidade que representa a norma é a Konnex Association.

Esta norma é baseada na tecnologia bem sucedida do EIB e alargada com os mecanismos de configuração e os meios físicos do BatiBUS e EHS.

O KNX é dos protocolos que mais meios de comunicação suporta. Estes são:

- Pares entrançados (*twisted pair*) a 4800 bits/s;
- Pares entrançados (*twisted pair*) a 9600 bits/s;
- Rede eléctrica a 110 kHz;
- Rede eléctrica a 132 kHz;
- RF a 868 MHz;
- Ethernet.

Alguns destes modos de comunicação são herdados das especificações anteriores e continua-se a dar-lhes suporte para uma maior compatibilidade entre sistemas [Weinzierl, 09] [KNX, 09].

### 1.3.2- X10

É a norma mais comum nos EUA para residências. Esta é uma norma internacional aberta para a indústria que especifica a comunicação entre dispositivos na casa. O meio de comunicação principal é a rede eléctrica (PLC - *Power Line Communication*), mas pode ainda comunicar por RF (a 310 MHz nos EUA ou a 433 MHz na Europa) para poder suportar dispositivos sem fios. Os pacotes transmitido tanto por RF como por rede eléctrica são muito semelhantes seguindo o formato X10.

Embora muito utilizada, esta norma tem algumas limitações. As comunicações através da rede eléctrica apresentam falhas quando alguns dispositivos domésticos estão a operar como fornos ou motores. Estes problemas podem ser minimizados com recurso à instalação de repetidores e outros componentes eléctricos. Outro problema conhecido do X10 é as interferências que dois sistemas vizinhos causam um no outro. É um protocolo lento, que apenas suporta 256 endereços para os dispositivos e não seguro. Apesar de continuar a ser muito usado, já é antigo (desenvolvido em 1975).

Uma das grandes vantagens deste protocolo é o facto de estar tão implementado que existem imensos dispositivos disponíveis no mercado com diversas funcionalidades. Como é um protocolo aberto qualquer fabricante pode produzir um dispositivo, desde que seja compatível.

Como exemplo dos dispositivos X10 apresentam as seguintes figuras.



**Figura 1.6** – Lâmpada X10, extraído de [EuroX10, 09]



**Figura 1.7** - Módulo X10, extraído de [EuroX10, 09]

Na Figura 1.6 podemos observar uma lâmpada com um módulo X10 integrado no casquilho (LM15ES). Recebe assim os comandos através da rede eléctrica (cada um dos módulos X10 sabe reagir a um certo número de comandos). A Figura 1.7 apresenta um

módulo X10 que controla aparelhos eléctricos ligados a tomadas de parede (AM12G). Recebe comandos X10 para ligar ou desligar o aparelho que lhe está associado.

Já a Figura 1.8 mostra um conversor de sinais X10 RF em comandos X10 para a rede eléctrica, repetindo assim o comando recebido por RF na rede eléctrica (S4022). Apesar desta função, ele próprio é um actuador X10. Quanto à Figura 1.9, esta apresenta um comando universal com ecrã táctil e luz de fundo azul, principalmente vocacionado para o controlo de dispositivos X-10 via RF e infra-vermelhos (IrDA) para os dispositivos normais.



**Figura 1.8** – Conversor X10, extraído de [EuroX10, 09]



**Figura 1.9** – Comando X10, extraído de [EuroX10, 09]

### 1.3.3- INSTEON

A tecnologia INSTEON utiliza dois meios de comunicação, a rede eléctrica e a RF. Os engenheiros da *SmartLabs, Inc* analisaram os protocolos já existentes e criaram uma solução mais robusta. Como o X10 é um protocolo amplamente utilizado decidiram suportá-lo pela rede eléctrica e conferir robustez e comodidade aos utilizadores através da comunicação RF. Os dispositivos INSTEON transmitem os comandos pela linha eléctrica e ainda por RF, formando assim uma rede. Ao contrário de outras soluções como Zigbee ou Z-Wave, devido à sua complexidade nos algoritmos de encaminhamento, a rede INSTEON implementa uma rede em malha dupla. Nesta rede todos os nós repetem as mensagens que recebem de modo a obter longos alcances sem ter que distinguir nós que reencaminham de outros de funções reduzidas.

Esta tecnologia, criada em 2001, é das mais robustas e fiáveis, suportando X10 e com todas as vantagens de uma rede em malha por RF [INSTEON, 09].

### 1.3.4- ZigBee

ZigBee é um protocolo desenvolvido pela *ZigBee Alliance* (composta por cerca de 300 companhias) de muito baixo custo, com baixos consumos para comunicações sem fios. Esta norma tem como objectivo ser utilizada nas áreas da automação de lares e edifícios, controlo industrial, periféricos para PC, aplicações sensoriais médicas, brinquedos, entre outras.

O ZigBee é baseado na norma IEEE 802.15.4 que define as especificações da camada MAC e camada física (PHY) para redes sem fios de área pessoal de baixo ritmo (Low-Rate WPANs).

O ZigBee está estruturado em camadas tal como o bem conhecido modelo OSI (Open Systems Interconnection).

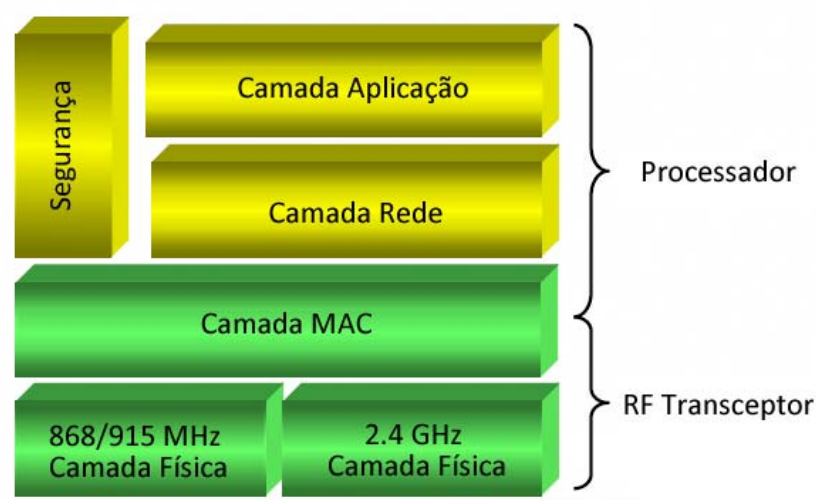


Figura 1.10 – Pilha de camadas ZigBee, adaptado de [Masters 08]

Na Figura 1.10 observam-se as duas camadas mais baixas, a camada MAC e a Física. Estas são descritas pela norma IEEE 802.15.4, e que muitas vezes vêm embutidas nos transceptores (*transceivers*) disponíveis no mercado. A camada de Rede e os mecanismos de encriptação são descritos pelo protocolo ZigBee. Muitos dos fabricantes destes componentes disponibilizam a *Stack* ZigBee elaborada (por exemplo a Mirochip e a Texas Instruments) de forma a simplificar a vida dos engenheiros. Já a



camada de aplicação é elaborada livremente pelos programadores conforme os requisitos do sistema.

Outro aspecto importante é que o ZigBee tem uma grande preocupação quanto à poupança de energia, visto a grande maioria dos seus dispositivos operarem a baterias. Já o Bluetooth é um protocolo que não tem essa preocupação e muito rapidamente gasta a energia de uma bateria (como se sabe pela experiência nos telemóveis).

O ZigBee define três tipos de nós de rede:

- Coordenador ZigBee (*ZigBee Coordinator*) (FFD)
- Encaminhador ZigBee (*ZigBee Router*) (FFD)
- Dispositivo Terminal ZigBee (*ZigBee End-Device*) (RFD)

Estes três tipos de nós podem ser agrupados em duas classes: os dispositivos com todas as funções (*Full Function Device* - FFD) e os dispositivos de funções reduzidas (*Reduced Function Devices* - RFD).

Os FFDs são dotados de todas as funcionalidades, estão sempre activos para receber mensagens e estão habitualmente ligados a uma fonte de energia. Já os RFD passam a maior parte do tempo adormecidos para poupar a energia das baterias. Existe a possibilidade de o seu nó parente (FFD) armazenar mensagens destinadas ao RFD, e quando este acorda requisita as suas mensagens.

Deste modo o Coordenador ZigBee é um FFD mas que actua como coordenador e executa funções muito específicas.

Podem ainda existir mais dois tipos de FFD, os simplesmente FFD e os FFD encaminhadores. A diferença reside nas topologias de rede apresentadas anteriormente. Na rede em árvore (*cluster tree*) os FFDs simples não reenviam mensagens, enquanto que nas redes em malha (*mesh*) todos os FFDs reenviam.

Já os dispositivos terminais ZigBee são sempre os RFD.

Nó parágrafo seguinte faz-se uma distinção do significado, usado até então, da palavra dispositivo. Assim, dispositivo (tradução de *device*) representa algo, quer seja

um sensor ou um actuador, que um nó da rede ZigBee possa ter. Por exemplo um nó que tenha um sensor de temperatura, e actue sobre uma lâmpada. Este nó contém dois dispositivos, embora do ponto de vista do *hardware* seja um objecto único.

Para cada dispositivo ZigBee tem de haver um perfil associado, isto é, se um nó da rede tem a capacidade de actuar sobre uma lâmpada, este nó tem um dispositivo e há forçosamente um perfil para este. Para os outros nós poderem interagir com o dispositivo daquele nó, estes sabem que o nó é de um tipo específico conhecendo assim quais os serviços que esse dispositivo oferece e como o controlar.

Este mecanismo é semelhante ao do USB (*Universal Serial Bus*). Quando introduzimos um dispositivo USB (de agora em diante a palavra dispositivo tem o significado normal) num PC, este tem que ter um perfil de utilização (*driver*) para o saber controlar e que serviços disponibiliza [ZigBee, 06] [IEEE, 06], [MASTERS, 08].

Uma das controvérsias desta tecnologia é sobre a interferência que é causada quando trabalha junto de dispositivos Bluetooth e redes WLANs devido ao facto de todas estas tecnologias operarem na mesma banda dos 2.4GHz. Existem trabalhos que estudam estes efeitos como por exemplo [Shin et al, 07].

Um dos entraves da propagação do ZigBee é os custos que lhe estão associados. Para utilização comercial do ZigBee é necessário ser membro da ZigBee™ Alliance. Para ser membro é preciso pagar US\$ 9.500 ou US\$ 3.500 conforme o estatuto do membro. O licenciamento de cada produto que uma empresa venha a desenvolver tem um custo muito elevado.

Deste modo alguns dos fabricantes de *hardware* decidiram simplificar o ZigBee e criar um protocolo proprietário, sem custos, para os sistemas que não necessitem de tamanha complexidade e robustez como a do ZigBee. Para a utilização destes protocolos proprietários é apenas necessário utilizar o *hardware* por ele fornecido. É o caso do MiWi da Microchip, do EZMac da Integration (que recentemente foi comprada pela Silicon Labs) ou do BeeKit da Freescale.

### **1.3.5- Outras Tecnologias**

Existem outras soluções para criar sistemas domóticos. Alguns são protocolos abertos, outros são soluções proprietárias. A título de referência apresentam-se algumas soluções que não serão aprofundadas nesta dissertação.

- LonWorks da Echelon
- Z-Wave da Z-Wave Alliance
- C-Bus ou SquareD Clipsal da Clipsal Integrated Systems
- HomePlug da Powerline Alliance
- EnOcean
- HAI da Home Automation Inc
- ModBus da Modicon
- ONE-NET
- Domintell
- Cardio
- iDom

### **1.4- Domótica em Portugal**

“Hoje a Domótica não é uma novidade em Portugal! E a prová-lo está a existência de várias empresas a operar no ramo, com a representação e distribuição de diversas marcas internacionais, especificamente dedicadas à Domótica” [Chamusca, 06].

Caminha-se “cada vez mais ao encontro dos conceitos fundamentais da domótica. Associado ao conceito do comando e controlo dos diversos níveis de conforto em casa, a palavra Domótica passou a fazer parte do vocabulário do português moderno.” [Chamusca, 06].

Alguns construtores portugueses começam já a preparar as habitações para os requisitos das instalações dos sistemas domóticos. “Na actual conjectura do ramo da construção, a Domótica surge como o maior e melhor valor acrescentado das

promoções imobiliárias, sendo as soluções dimensionadas e custeadas mediante o alvo pretendido” [Chamusca, 06]. Alguns empreendimentos novos ou em fase de construção já estão dotados de sistemas domóticos.

As empresas de Domótica indicam uma subida da procura na ordem dos 40%, apesar de estas habitações continuarem a ser uma minoria devido aos preços envolvidos. A construção preparada para a domótica é um factor de diferenciação no preço final. Os preços variam muito conforme as funcionalidades que se querem conferir à casa e às suas dimensões. Em 2007 a rede de pesquisa de Robótica previa para os próximos anos uma Europa com cerca de 15 milhões de casas inteligentes. Actualmente a massificação da inteligência doméstica, e a consequente baixa de preços, ainda é uma miragem, mas há especialistas que afirmam que estas serão a casa comum dentro de uma década.

Por enquanto estes sistemas são mais impulsionados pela segurança e pela poupança energética, uma vez que o conforto é muitas vezes visto como supérfluo na nossa sociedade.

Um factor que vai contribuir bastante para a disseminação destes sistemas em toda a Europa é a Directiva Europeia 2002/91/CE sobre o Desempenho Energético dos edifícios. Esta directiva exige que os edifícios sejam energeticamente eficientes. Todos os edifícios novos vão ter obrigatoriamente um certificado que atesta o nível de poupança de energia. Embora esta directiva seja de 2002, em 2006 apenas quatro países da União Europeia (UE) a seguiam. Prevê-se que ela esteja em fase de implementação na maioria da UE em 2009.

Para que um edifício seja energeticamente eficiente a qualidade da construção é fundamental mas a domótica pode ajudar e por isso prevê-se um aumento significativo da instalação dos sistemas domóticos [SIC, 07] [RTP, 08] [Maldonado, 06].

Em Portugal, no Museu das Comunicações, está desde Maio de 2003, em exposição a Casa do Futuro Inclusiva onde é apresentada uma solução complexa de automação doméstica, que utiliza um elevado conjunto de tecnologias de ponta, num ambiente seguro, lúdico, confortável e moderno.

Actualmente o sistema KNX é o mais utilizado na Europa e consequentemente em Portugal. Este é implementado por mais de uma centena de fabricantes na Europa e na Ásia.

Existem alguns trabalhos de investigação portugueses nesta área. Por exemplo no trabalho em [Gomes, et al, 97] e [Gomes, et al, 98] é apresentada uma metodologia para a modelação de sistemas aplicados em edifícios inteligentes. Já em [Conceição, 04] é proposta uma metodologia e arquitectura para a monitorização e vigilância de edifícios inteligentes.

Em [Nunes, 03] e [Nunes, 04b] surge uma proposta de protocolo de comunicação para edifícios inteligentes o DomoBus. Em [Nunes, 05] é apresentada uma concretização de uma rede extremamente simples e de muito baixo custo, com aplicações na área da automação de edifícios. Este trabalho tem vários aspectos em comum com o desenvolvido e apresentado nesta dissertação. Já em [Nunes, 06] é proposta uma arquitectura descentralizada para a supervisão de sistemas de automação doméstica.

Na investigação têxtil surge uma inovação desenvolvida na Universidade da Beira Interior. Chamado de DomoVest, é um “casaco domótico” com a capacidade de accionar dispositivos domóticos através de um teclado todo elaborado em tecido (ligado através de fios a um transmissor). Este foi apresentado no MOVE 08 – Mostra de trabalhos dos Cursos de Design de Moda.

Existe também um projecto iniciado pela AveiroDomus que se propôs a edificar a casa do futuro. Esta casa chamada InovaDomus que será realizada no Campus da Universidade de Aveiro, pretende repensar todo o conceito de habitat e dotar o espaço com as mais inovadoras soluções tecnológicas disponíveis. Esta casa servirá como um laboratório de “tecnologia habitável”.

Quanto ao mercado actual da Domótica em Portugal já existem bastantes empresas a operar e inovar.

No mercado de *software* de automação, segurança e entretenimento em casa, destaca-se o desenvolvimento da empresa QuiiQ que apresenta soluções de software capazes de gerir toda a domótica de uma casa, com uma forte componente

Multimédia (baseado no Windows Media Center). Para isso estas soluções suportam alguns protocolos conhecidos, referidos anteriormente [SIC, 08] [EXAME, 08].

Quanto a empresas a instalar sistemas domóticos ou a revender componentes actualmente já existem bastantes. Por exemplo, EuroX10, DOMUS CONNECT, JG Domótica, CentralCasa, DreamDomus, LogicHome, Domática, Hidomus entre muitas outras. Outro exemplo é a Legrand que disponibiliza dispositivos de fácil montagem para uma casa inteligente. A PT Comunicações tem ainda disponíveis pequenos kits (PowerMax e HomeGuard) com funcionalidades tipicamente domóticas como “Telesegurança” ou “Televigilância”.

## 2- Sistemas Embutidos em Rede

Embora não directamente relacionadas com a área específica da Domótica as WPAN (Wireless Personal Area Network) e as WSNs são temas que merecem constar nesta dissertação. As metodologias, o hardware, o software, os protocolos e muitas outras questões são partilhadas entre a domótica usando as RF, as WPAN e as WSN.

Estas três temáticas fazem parte da área de investigação ligada às redes de sistemas embutidos (*Networked Embedded Systems* - NES) sem fios. A grande diferença entre estes três domínios é fundamentalmente a área de aplicação. Como se sabe a domótica utiliza as redes sem fios por RF em função de automatismos domésticos.

Já as WPAN são usadas para interligar dispositivos pessoais tais como telemóveis, assistentes pessoais digitais (*Personal Digital Assistant* - PDA), impressoras, máquinas fotográficas, sistemas de posicionamento global (*Global Positioning System* - GPS), comandos de consolas, brinquedos, entre outras aplicações. As WPAN surgiram principalmente com o Bluetooth.

Por sua vez, as WSN têm finalidades bem diferentes como aplicações militares, meteorológicas ou científicas.

Assim sendo, estas três áreas partilham o mesmo domínio científico e tecnológico, embora com fins diferentes. Em seguida apresenta-se uma caracterização sobre alguns temas inerentes à área das redes de sistemas embutidos (NES) sem fios, e em particular às WSNs.

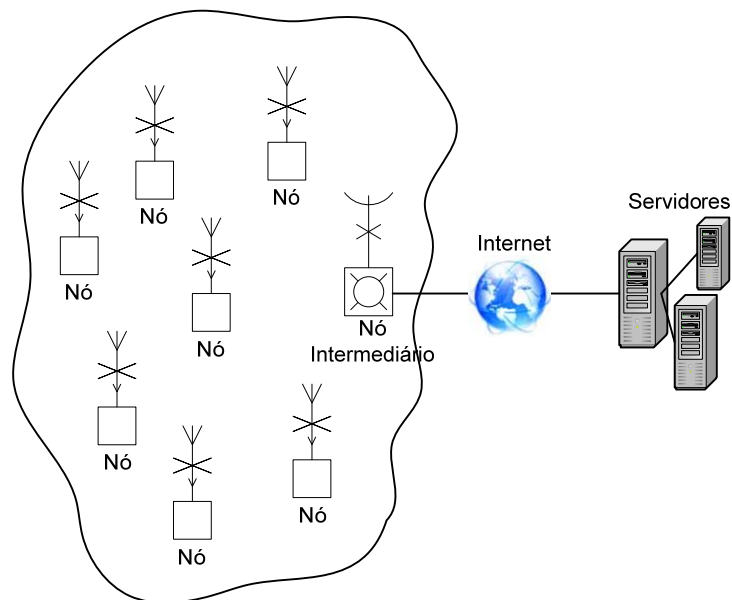
Esta caracterização é muito importante uma vez que o trabalho desenvolvido nesta dissertação envolve muitas destas questões e pode, com muita facilidade, migrar e ser aplicado à área das WSN. Isto porque numa visão muito simplista, e excluindo os elementos multimédia, a domótica pode ser vista como um conjunto de sensores e actuadores que reagem a certos estímulos. Nesta visão, a domótica não será mais do que uma WSN com a capacidade de reagir e actuar (há quem considere que as WSN também podem ter actuadores, e há ainda quem considere que os seus nós não só actuam, como podem ser actores na decisão, sendo designadas de redes sem fios de sensores e actores (*Wireless Sensor and Actor Network* - WSAN) [Akyildiz – Kasimoglu, 04]). Claro que esta visão da domótica é muito minimalista, relata praticamente os

primeiros sistemas para os edifícios inteligentes e exclui todos os interfaces e mecanismos automatizados/inteligentes que a Domótica actual fornece.

## 2.1- Introdução

As redes de sistemas embutidos surgiram na última década com os rápidos avanços da microelectrónica na área dos microcontroladores que integrados com rádios ou através de ligações por cabo na mesma placa formaram os elementos da rede. Estas redes são tipicamente constituídas por sistemas embutidos que têm que comunicar uns com os outros formando uma rede. Cada sistema é um elemento/nó da rede. Cada elemento deve interagir com os outros elementos, mas ainda com o meio envolvente, através de sensores e/ou actuadores [Gupta et.al, 06].

WSN é o nome genérico no qual um amplo grupo de dispositivos se agrupa. Qualquer dispositivo equipado com um processador, com a capacidade de “sentir” (através de sensores), comunicar sem fios, e ainda com a habilidade de se associar e organizar com outros de modo a formar uma rede pode-se classificar um elemento de uma WSN [Dulman et.al, 06].



**Figura 2.1** – Topologia típica WSN

Na Figura 2.1 mostra-se uma topologia típica das WSN. Nesta topologia diversos nós monitorizam uma certa área (representada pela linha que envolve os nós), e vão



recolhendo informações. Em certa altura do seu funcionamento estes transmitem a informação recolhida que viaja de nó em nó por RF, até ao nó intermediário (*gateway*) que está ligado a outra rede, por exemplo internet, e transfere a informação para um servidor que aloja e processa os dados recolhidos.

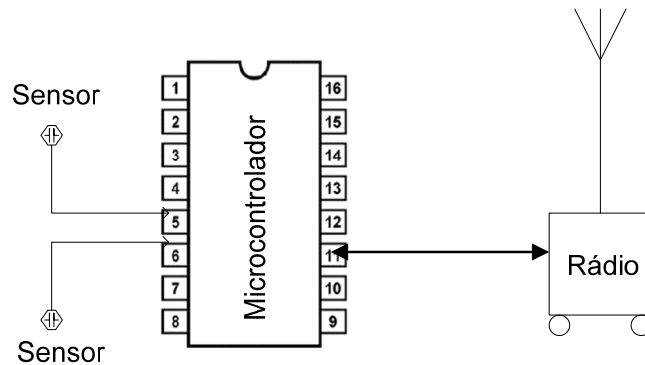
Estes nós, também conhecidos como *motes*, são tipicamente de dimensões reduzidas e relativamente baratos. São colocados na zona em que existe algo de interesse que os sensores irão monitorizar. Nas aplicações actuais destas redes os nós são estacionários mas podem perfeitamente ser montados em objectos móveis. O facto de estes serem embutidos, terem a capacidade de monitorizar através de sensores e terem a aptidão de comunicar define esta área de redes de sensores e faz a separação entre outras áreas como monitorização remota, computação móvel através de computadores portáteis, ou mesmo de sistemas de monitorização centralizada.

Estudos nesta área remontam a 1990, mas é por volta de 2004 que se torna economicamente viável a implementação destes sistemas. Desde então esta tem sido uma área com um enorme interesse no campo da investigação, quer académica quer comercial e até militar. Grande parte desta investigação tem-se focado no consumo e gestão dos recursos energéticos, nos processos de aquisição de informação pelos sensores, no modo de actuação, e ainda como gerir estas redes com imensos elementos, que entram e saem da rede a qualquer momento e fornecem serviços diferentes entre eles [Heidemann-Govindan, 05]. Existe ainda a aplicação das WSN mas em tamanhos microscópicos chamados de SmartDust. Estas redes são formadas por sistemas microelectromecânicos (microelectromechanical systems - MEMS) de tamanhos à escala nanométrica provenientes da nanotecnologia.

## 2.2- Hardware

Os elementos (*motes*) têm a necessidade de ter uma unidade central de processamento (Central Processing Unit - CPU) de funções genéricas, memórias e de pinos de entrada e saída para interagir com os sensores e periféricos. Deste modo é

necessário um microcontrolador. Então os *motes* são formados essencialmente por um microcontrolador, sensores e um rádio transceptor, como mostrado na Figura 2.2.



**Figura 2.2** – Esquema de um *mote*

Qualquer microcontrolador que preencha os requisitos pode ser utilizado, como por exemplo os da Microchip, Silicon Labs, Atmel, Texas Instruments, Freescale, entre muitos outros. Na escolha deste microcontrolador é preciso ter como factor decisivo os seus consumos. Estes são microcontroladores de muito baixo custo.

Relativamente aos transceptores há bastante oferta no mercado. Todos os fabricantes de microcontroladores mencionados acima disponibilizam também estes transceptores. Existem ainda soluções que juntam tudo num objecto único, por exemplo os SoC (System-on-Chip) da Ember. Em [Varchola-Drutaroský, 07] e [Körber et.al, 05] é apresentado um quadro comparativo destas soluções.

No trabalho desenvolvido nesta dissertação foram utilizados os microcontroladores PIC24F128GA010 e PIC18F4620 da Microchip, e os transceptores foram os MRF24J40 da Microchip também.

Os sensores acompanham a evolução tecnológica e a miniaturização dos sistemas. Com a proliferação de sistemas MEMS muitos sensores foram incorporados nas plataformas dos nós (*motes*). Existem no mercado muitos fabricantes que disponibilizam pequenos sensores com a tecnologia MEMS. Muitos destes fabricantes estão segmentados conforme a área de aplicação. Apenas alguns estão centrados na área das WSN, por exemplo a Ember e Millenial Net [Heidemann-Govindan, 05].

## 2.3- Software

A parte do software tem sido profundamente estudada, pois é nela que se tenta resolver os maiores desafios destes sistemas. Por exemplo, pretende-se modelar um comportamento que não desgaste as baterias, com períodos de adormecimento do *hardware* bem geridos, que através dum protocolo robusto mantenha uma rede complexa e ainda que tenha a capacidade de interagir com o ambiente envolvente.

A gestão da rede é muito importante porque mantém uma ligação vital entre todos os elementos de modo a permitir a sua colaboração entre eles. Sabe-se que o rádio é dos componentes destes sistemas que mais energia consome, entre 20% a 40%. Deste modo uma boa gestão de rede, através de protocolos optimizados, é de extrema importância no prolongamento da vida de um elemento, e da própria WSN.

A camada MAC está directamente ligada aos consumos energéticos. A colisão de pacotes despende energia ao originar retransmissões. Ter o transceptor activo à escuta de potenciais mensagens, ou estar constantemente a acordar o microcontrolador devido a mensagens que não lhe são destinadas resultam num gasto energético. A norma IEEE 802.11 conhecida como “wi-fi” utiliza CSMA (Carrier-Sense Multiple Access) que escuta a portadora antes de transmitir de modo a evitar colisões. Outro método para gestão das transmissões é o TDMA (Time-Division Multiple Access) que divide o tempo em vários intervalos (*slots*) em que cada nó transmite no seu intervalo específico, não se sobrepondo a mais nenhuma transmissão. Como as WSN são redes em que a informação a transmitir é de tamanhos reduzidos foi criada uma norma para redes sem fios de baixo ritmo, a IEEE 802.15.4, especificamente para sistemas deste tipo.

Baseada nesta norma surgiram entretanto vários protocolos na camada de rede como o ZigBee e outras variações apresentadas anteriormente na secção 1.3. Estes protocolos especificam como a rede é formada, o formato das mensagens, quando se efectuam transmissões, como reencaminhar mensagens para outros nós, como sincronizar todos os nós, entre muitas outras funcionalidades [Heidemann-Govindan, 05].

De modo a facilitar a interligação das tarefas de rede, de aquisição de informação do meio envolvente e ainda processar tudo isto com recursos tão limitados como os

disponíveis num microcontrolador foram criados Sistemas Operativos dedicados (Operating System - OS), como por exemplo o eCos, o LynxOS e TinyOS. Estes OSs criam um núcleo que gere o encadeamento do processamento, conferindo tempos de execução às tarefas (*threads*) que o sistema tem a fazer. O mais conhecido e implementado na área das redes de sistemas embutidos é o TinyOS que começou como um projecto na Universidade da Califórnia, Berkeley, foi crescendo e actualmente é formado por uma grande comunidade de investigadores e entusiastas da área.

O TinyOS [TinyOS, 09] baseia-se no princípio de que a maior parte das redes de sensores são baseadas em eventos (*event-driven*), ou seja o sistema reage a eventos, por exemplo captados nos seus sensores. É composto por uma estrutura com vários componentes. Cada componente é um módulo de *software* que fornece uma abstracção dos processos internos de cada módulo, seja esse o controlo de um dispositivo de *hardware* ou alguma tarefa de como processar informação. Tipicamente em resposta a um evento captado pelos sensores ou alguma interrupção do *hardware*, são chamadas uma cadeia de componentes/módulos que respondem a esse evento.

Na abstracção implementada pelo TinyOS são fornecidos duas funcionalidades: a tarefa, que fornece a cada módulo tempo de processamento enquanto o processador estaria em espera (*idle*), e o comando, que manda um módulo executar uma função específica (tal como enviar uma mensagem). Através destas abstracções é possível modelar graficamente a aplicação. Ao enfatizar a programação baseada em eventos (*event-driven*) em vez de multi-tarefas (*multi-threading*), permite que os recursos de *hardware* sejam menores. Esta ferramenta permite a reutilização de código, melhora a modularidade do sistema através dos componentes, e minimiza o tamanho do programa que, em dispositivos limitados, é crucial.

A programação para este OS é feita através da linguagem nesC que contém as estruturas necessárias à abstracção do TinyOS: componentes, tarefas, comandos e eventos. O nesC fornece vários mecanismos para otimizar o programa para esta plataforma [Heidemann-Govindan, 05] [Gupta et.al, 06].

Muitos trabalhos estão a ser desenvolvidos com estas ferramentas, em especial na Universidade da Califórnia, Berkeley. Um desses trabalhos é o galsC.

O galsC é a linguagem para o TinyGALS, um modelo globalmente assíncrono localmente síncrono (GALS). O modelo TinyGALS fornece métodos de implementar tarefas concorrentes, chamadas de actores. No nível da aplicação os actores comunicam uns com os outros assincronamente através de mensagens. Dentro de cada actor os componentes comunicam sincronamente através da chamada de métodos. Assim o modelo de programação em termos de fluxo de controlo é globalmente assíncrono e localmente síncrono.

Este modelo tem por base o TinyOS, mas ao contrário do TinyGALS as tarefas concorrentes no TinyOS não são acessíveis como parte do interface de um componente. Falha na gestão explícita da concorrência e obriga os programadores de componentes a utilizarem semáforos na modelação da concorrência [Cheong et al, 03] [Cheong-Liu, 04]. Mais à frente, nesta dissertação, vai-se abordar o conceito GALS com maior detalhe.

Estes sistemas NES têm ainda outras dificuldades que continuam a ser estudadas.

Uma vez que os nós (*motes*) podem não ser colocados nas suas posições finais, como largados de um avião, ou terem mobilidade, é de extrema importância saberem onde se situam. Caso contrário podemos estar a receber informação do meio envolvente, mas se não sabemos qual é esse meio a informação é inútil. Assim actualmente existem algumas abordagens para a resolução desta questão. Uma delas é com o conhecimento da localização de alguns nós específicos, calculando a distância entre esses nós é possível fazer uma triangulação. Essa distância é calculada através da potência do sinal RF que chega ao receptor. Claro que existe alguma incerteza nestes cálculos, até porque a propagação varia conforme os locais e condições meteorológicas. Outras técnicas em estudo aplicam a tecnologia RADAR, ou através da contagem do numero de saltos (*hops*) de uma mensagem.

Outra dificuldade surge quando é necessária a sincronização temporal entre os elementos da rede. Esta funcionalidade pode ser importante quando se pretende fazer uma amostragem colectiva, em que todos os nós devem recolher informação dos seus sensores ao mesmo tempo. Métodos para sincronização baseiam-se num carimbo temporal (*time stamp*) que é colocado em cada mensagem de modo a que o receptor se sincronize. Deste modo toda a rede se vai sincronizando à medida que trocam

mensagens. Mais uma vez surge o problema que o tempo de propagação no ar não é constante, mas também o processamento das mensagens e o tempo de transmissão e recepção podem introduzir atrasos.

À medida que os nós fazem as suas leituras vão gerando informação. Esta informação, quando partilhada com a dos outros nós, permite prever algumas situações. A questão então é onde guardar tanta informação gerada. Existem soluções como o TinyDB ou o The Cougar Sensor Database que permitem armazenar informação de uma forma reduzida.

Outro desafio a estes sistemas é como fazer uma programação remota. Esta reprogramação pode ser efectuada através de parâmetros que ao serem alterados fazem com que o funcionamento normal mude para outro previamente implementado, o que pode acontecer quando se anteciparam certas necessidades. Outros métodos baseiam-se na alteração total ou parcial do programa que o nó tem.

Em relação à segurança nas WSN esta ainda não é muito forte. São alvos fáceis de ataques de negação de serviço (*denial of service*). Existem mecanismos de encriptação e autenticação que são utilizados para a confidencialidade das mensagens. Esta encriptação pode exigir mais processamento o que resultará num maior consumo energético. Conforme as aplicações, a segurança pode ser ou não necessária [Heidemann-Govindan, 05] [Gupta et.al, 06] [Dulman et.al, 06].

## **2.4- Aplicações**

Como já vem sido referenciado ao longo desta dissertação as WSNs têm bastantes aplicações. Em seguida apresentam-se algumas delas.

### **Aplicações Militares**

As aplicações militares têm sido um dos catalisadores das WSN. Muito do desenvolvimento WSNs deve-se a esta área de aplicação onde podem ser utilizadas

para diversas funcionalidades. A monitorização de áreas ou estradas, de modo a detectar intrusos ou veículos é um exemplo. Outro exemplo é para obter informação sobre um campo de batalha ou uma área inimiga; um avião passa largando vários *motes* que vão recolhendo informações do terreno e da actividade do inimigo. Espalhando *motes* num campo de batalha pode-se, na altura do combate, inibir as comunicações do inimigo desorientando-o. Podem ainda ser usadas para a detecção ou marcação de alvos a abater pela aviação, detecção de armas nucleares ou biológicas, entre outras aplicações.

### **Aplicações Ambientais**

Um exemplo é a monitorização de habitats. Colocando nós nas tocas ou refúgios dos animais é relativamente fácil monitorizar parâmetros como temperatura, humidade, luz, sons e os horários do quotidiano dos animais. Esta monitorização é feita de um modo não invasivo e durante longos períodos de tempo. Outra aplicação que pode ser feita é a monitorização da presença e movimentação de certas espécies numa área.

Na biologia marinha prevê-se a utilização das WSN para a monitorização de certas zonas. Com os nós colocados a uma pequena profundidade, estes captariam dados relativamente à temperatura da água, luz e salinidade, factores que são importantes à formação de grandes colónias de algas.

Outra aplicação que se prevê é a monitorização de solos ou águas de modo a evitar contaminações em zonas industriais. É aplicável também na detecção de fogos florestais, entre outras.

### **Aplicações Cívicas**

Utilizando as WSN é possível monitorizar pessoas idosas ou com alguma doença crónica tanto em casa como no Hospital. Em casa podem monitorizar os movimentos do paciente, alertar para a toma de medicamentos, detectar alguma queda ou ainda activar alarmes quando os dados vitais do paciente saírem de certos limites. Por sua vez no Hospital as WSN permitiriam localizar facilmente médicos ou pacientes e continuar a monitorizar os dados vitais do paciente.

No domínio das habitações existem inúmeras aplicações já abordadas anteriormente.

Na monitorização de edifícios as WSN podem avaliar o estado do edifício e prevenir para uma eventual derrocada ou acidente grave, através da monitorização das paredes, fundações e juntas de dilatação do edifício, como é feito nas barragens.

Outra aplicação é na prevenção de catástrofes. Por exemplo os sismólogos podem monitorizar a propagação das ondas sísmicas em diferentes áreas ao pormenor, e até avaliar a reacção dos edifícios a estas e detectar falhas.

### **Aplicações Industriais**

Na área industrial as WSN podem ser utilizadas por exemplo numa linha de montagem para monitorizar o estado de equipamentos ou as condições de trabalho em termos do ambiente na zona de laboração. Outras aplicações surgem na área da segurança e monitorização de objecto, por exemplo em museus ou galerias de arte. [Heidemann-Govindan, 05] [Gupta et.al, 06] [Dulman et.al, 06].



### 3- Formalismos de Modelação

A importância da fase de especificação de um sistema pode-se relacionar com complexidade do sistema a definir. Os sistemas embutidos encontram-se cada vez mais complexos devido a diversos factores, entre eles: a maior capacidade dos dispositivos permitindo um aumento dos sistemas, a necessidade de desempenhos exigentes, a fiabilidade dos sistemas, as restrições relativamente a consumos energéticos, os custos e o tempo de desenvolvimento até chegar ao mercado (*time-to-market*).

No sentido de simplificar e melhorar a fase de especificação, têm sido utilizados diversos formalismos de modelação. Tipicamente estes formalismos especificam o comportamento do sistema de uma forma precisa e baseiam-se em modelos matemáticos.

Os formalismos de modelação são ainda designados por Modelos de Computação (MdC). Estes são constituídos por uma notação e pelas regras de computação do comportamento. [Gomes et al, 06].

A utilização destes modelos em sistemas embutidos permite, pelo menos, um dos seguintes aspectos:

- Percepção dos requisitos comportamentais do sistema sem qualquer ambiguidade.
- Verificação e validação das especificações funcionais consoante as funcionalidades pretendidas.
- Suporte à síntese de uma arquitectura específica e às suas interacções.
- Possibilidade de utilização de várias ferramentas baseadas no mesmo modelo permitindo a produção do sistema por ferramentas e pessoas diferentes. [Sgroi, 00]

Assim a verificação das propriedades de um sistema através de modelos é muito importante no projecto de sistemas embutidos (e mesmo na generalidade dos sistemas). Esta permite a validação do sistema, assim como a sua viabilidade (a especificação do sistema pode ser fisicamente impossível) e ainda fornece uma noção de dificuldade e custos do desenvolvimento do mesmo.

A construção do modelo de um sistema fornece diversas vantagens e garante uma compreensão mais completa de todo o sistema, tornando assim mais fácil a identificação

das propriedades do sistema, tanto as pretendidas como as não pretendidas conforme os requisitos.

Embora haja trabalhos que defendam o contrário, como o apresentado em [Dijkstra, 92], a grande parte dos formalismos de modelação para sistemas embutidos são baseados numa linguagem gráfica, sendo extremamente úteis e populares no desenvolvimento de software. Um dos mais utilizados é o *Unified Modeling Language* (UML) que será abordado mais a frente na secção 3.3 desta dissertação.

As representações gráficas de máquinas de estado finitas, quer de Moore quer de Mealy, são bons exemplos destes formalismos gráficos. Outro exemplo são as Redes de Petri (RdP) que já permitem controlo de tempo, comunicação e ainda concorrência.

Havendo uma grande variedade de formalismos de modelação, existem diversas possibilidades, mas a escolha de um formalismo específico pode ser uma tarefa árdua. Diferentes sistemas embutidos podem focar diferentes aspectos, como por exemplo restrições de tempo-real, a capacidade de reagir a um evento ou mesmo a necessidade de interagir com outros sistemas circundantes. Deste modo a escolha do formalismo específico deve ter em conta a capacidade de cada um para modelar certos aspectos comportamentais.

Existem vários MdC para sistemas embutidos com diferentes topologias, por exemplo existem MdC que são focados no controlo (*control dominated*) sendo assim salientado o aspecto reactivo do sistema. Já outros sistemas focam-se no processamento de dados. [Gomes et al, 06].

Por vezes a heterogeneidade do sistema pode fazer com que um único formalismo não seja suficiente para modelar todo o sistema. Neste caso o processo indicado é decompor o modelo do sistema em vários submodelos e aplicar o formalismo indicado a cada um. Neste processo é ainda necessário integrar todos os submodelos de um modo coerente, para que estes consigam interagir entre si [Buck et.al, 94].

O tema dos formalismos de modelação tem sido amplamente estudado e existe bastante literatura sobre este assunto. Nas secções seguintes serão descritas alguns desses formalismos.

### 3.1- Fluxogramas

Os fluxogramas são talvez o formalismo de modelação mais antigo e mais utilizado desde sempre. São amplamente utilizados pelos programadores/desenhadores de sistemas no seu dia-a-dia, como método de explicar e esquematizar graficamente modelos de sistemas.

A sua simplicidade é uma das grandes vantagens, mas ao mesmo tempo em certos sistemas este formalismo pode ser insuficiente.

Em tom informal, os fluxogramas são a maneira de explicar qualquer mecanismo, funcionamento ou raciocínio que todo o Engenheiro Electrotécnico rápida e desenrascadamente utiliza (Eng. de Sistemas ou Eng. Informático). Quem nunca presenciou algo deste género desenhado rapidamente na toalha de mesa de um restaurante? O autor desta dissertação conhece quem (Eng. Electrotécnico) de modo a explicar o itinerário até ao local do seu casamento, o fez através de fluxogramas. Claro que este fluxograma foi a referência para pessoas de todas as idades, de diferentes áreas e graus de escolaridade.

Os Fluxogramas surgiram por volta de 1947 num trabalho desenvolvido na Universidade de Princeton, e muitos anos depois, com as naturais evoluções, viria a ser definida uma norma a especificar a sua utilização e representação, a ISO 5807,1985.

Os fluxogramas permitem apresentar graficamente o comportamento de sistemas através da representação do fluxo de controlo, do fluxo de dados e ainda das interacções que o sistema tem com entradas e saídas.

Esta representação é feita através de uma notação gráfica que se explica em seguida de uma forma resumida.



**Figura 3.1** - Terminal

**Terminal** – Representa o início e o fim do fluxo lógico do programa.  
Pode ainda ser usado na definição de subrotinas.

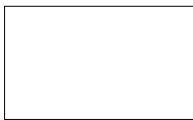


Figura 3.2 - Processamento

**Processamento** – Representa a execução de uma operação ou grupo de operações que estabelecem o resultado de um cálculo lógico ou matemático.

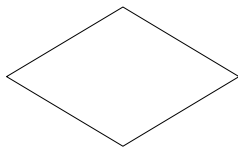


Figura 3.3 - Decisão

**Decisão** – Representa o uso de desvios condicionais para outros pontos do programa de acordo com situações variáveis. Avalia uma condição e o programa segue conforme o resultado.



Figura 3.4 – Entrada Manual

**Entrada Manual** – Representa a entrada manual de dados, normalmente efectuada num teclado ou por qualquer tipo de interface com o utilizador.

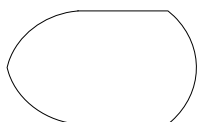


Figura 3.5 - Exibição

**Exibição** – Representa a execução da operação de saída visual de dados num monitor ou qualquer tipo de saída visual para o utilizador.

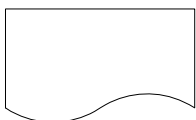


Figura 3.6 - Documento

**Documento** – Representa a execução da operação de saída de dados num documento, por exemplo através de uma impressora.



Figura 3.7 - Dados

**Dados** – Representa dados de uma forma genérica. Tipicamente é associado a operações genéricas de entrada e saída de dados, desde que identificados.



Figura 3.8 – Dados Gravados

**Dados Gravados** – Representa o acesso a um ficheiro guardado, tanto para leitura como para escrita.

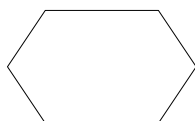


Figura 3.9 – Preparação

**Preparação** – Representa a modificação de instruções ou grupo de instruções existentes em relação à actividade subsequencial.

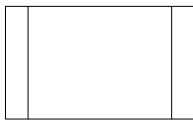


Figura 3.10 – Processo Predefinido

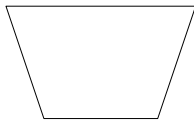


Figura 3.11 – Operação Manual



Figura 3.12 - Conector



Figura 3.13 - Linha

**Processo Predefinido** – Representa a definição de um grupo de operações estabelecidas como uma sub-rotina de processamento anexa ao diagrama de blocos.

**Operação Manual** – Representa a execução de qualquer operação que possa ser realizada pelos utilizadores do sistema.

**Conector** – Representa a entrada ou saída em partes do diagrama de blocos. Pode ser usado na definição de quebras de linha e na continuação da execução de decisões.

**Linha** – Representa a acção das relações existente entre os vários símbolos do diagrama. Normalmente possui a ponta de uma seta indicando a direcção do fluxo de acção [Manzano, 04].

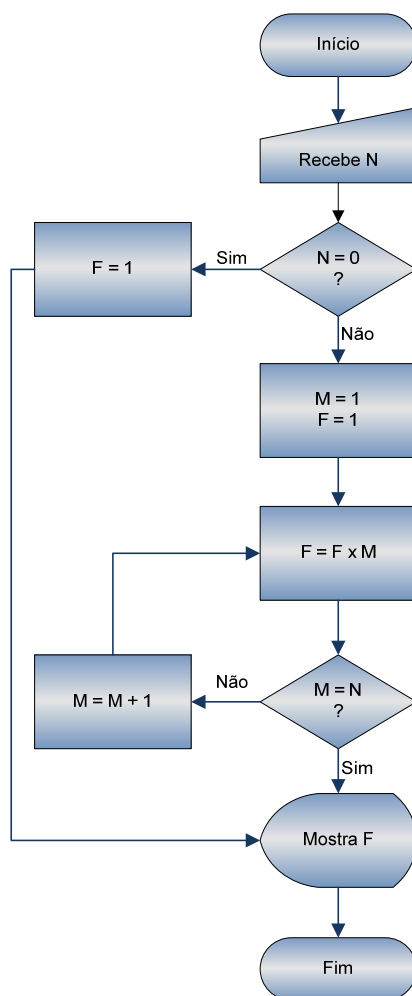


Figura 3.14 – Exemplo de Fluxograma

Na Figura 3.14 mostra-se o exemplo de um fluxograma para o cálculo da função factorial  $F = N!$ . Assim o programa após ter iniciado vai receber do utilizador o número  $N$ . Em seguida verifica se  $N$  é igual a zero, e se for, afecta  $F$  com o valor um e mostra ao utilizador. Se não, vai-se calcular o factorial do número  $N$ , inicializando  $M$  a um.  $N$  vai ser multiplicado por  $M$ ,  $N$  vezes até que  $M$  seja igual a  $N$ . Quando  $N$  é igual a  $M$ , o valor do factorial de  $N$  é  $F$ . Em seguida mostra-se ao utilizador o valor de  $F$  e o programa termina.

Notar que o símbolo em “Recebe  $N$ ” e “Mostra  $F$ ” podia ser trocado pelo símbolo Dados, que representa a entrada e saída de dados de forma genérica.

## 3.2- Redes de Petri

### 3.2.1- Enquadramento Histórico das Redes de Petri

O conceito das Redes de Petri surgiu em 1962 quando Carl Adam Petri apresentou a sua dissertação de Doutoramento, à Faculdade de Matemática e Física da Universidade Técnica de Darmstadt na Alemanha [Petri, 62] [Petri Nets World, 2007].

Esta não foi uma dissertação convencional, na medida em que não tentava resolver um problema específico, nem elaborar uma nova teoria. Em vez disso, Petri apresentou uma variedade de ideias e propostas acerca dos fundamentos da informática.

De modo a fundamentar as suas ideias inovadoras Petri apresenta um conceito de como resolver um problema muito conhecido na altura, a questão das funções recorrentes. Dada uma função  $f$  definida por recorrência, e  $n$  um argumento,  $f(n)$  pode exigir um maior número de recursos do que os existentes. Petri propõem uma solução através de uma arquitectura modular, em que cada módulo teria que funcionar independentemente, e todo o sistema funcionaria de modo assíncrono. Esta arquitectura idealizada por Petri tem muitos aspectos em comum com as arquitecturas GALS actuais, como se verifica mais à frente na secção 4 desta dissertação. O seu trabalho tinha como objectivo desenvolver um formalismo em que as máquinas de estado fossem capazes de comunicar entre si.

Pouco depois, Petri apresenta outro trabalho [Petri, 63] em que prova que a sua construção/arquitectura é computacionalmente universal.

Ao longo dos seus estudos Petri utiliza notações formais para sistemas distribuídos assíncronos, compostas por representações gráficas e fórmulas algébricas com o operador “paralelo”, que seria o que veio depois a resultar na álgebra de processamento (*process algebra*). Com a evolução desta representação gráfica acabou por resultar nas RdP, com os lugares e transições [Brauer-Reisig, 06]. As RdP como são conhecidas nos dias de hoje apareceram por volta de 1966 quando Petri as apresentou num Colóquio em Hannover [Petri, 67].

Foi assim criado um novo formalismo de modelação em que a característica principal era a possibilidade de representar concorrência.

De início foram pouco utilizadas, recebendo a partir do final da década de 60, impulsos significativos vindos de várias áreas de aplicação como os sistemas de manufactura e os protocolos de comunicação [CSD, 09] [Petri Nets World, 2009].

Embora as RdP fossem apresentadas apenas em 1966, já em 1964 eram utilizadas por um membro da *Bells Telephone Laboratories* [Brauer-Reisig, 06]. Em 1968 surge outra aplicação das RdP no projecto norte-americano *Information System Theory*, da A.D.R. (*Applied Data Research, Inc*) evidenciando assim a aplicabilidade das RdP na análise e na modelação de sistemas com componentes concorrentes [Heuser, 91].

Desde então, as RdP têm sido amplamente estudadas e desenvolvidas devido às suas numerosas potencialidades de modelação, tais como: sincronização de processos, concorrência, conflitos e partilha de recursos. Têm sido elaborados muitos trabalhos teóricos sobre as RdP, resultando num desenvolvimento e enriquecimento das mesmas, assim como novas técnicas de análise e ainda variantes do modelo inicial. Algumas tentativas de aplicação a novas áreas fizeram com que aparecessem extensões ao modelo inicialmente proposto, surgindo assim muitas classes de RdP.

Actualmente podem-se encontrar as RdP aplicadas em diversas áreas, como por exemplo [Marranghello, 05]:

- Automação de escritórios
- Automação de manufactura
- Avaliação de desempenho de sistemas
- Circuitos integrados (ICs)
- Protocolos de comunicação
- Sistemas distribuídos
- Sistemas de Produção

### **3.2.2- Conceito de Redes de Petri**

Como já referido as RdP são um formalismo de modelação/especificação de sistemas. São uma ferramenta matemática e gráfica através dum ambiente uniforme para a modelação, análise formal e simulação de sistemas a eventos discretos, permitindo uma visualização simultânea da sua estrutura e comportamento [Zurawski et al, 94].

Estas especificam dois aspectos dos sistemas, eventos e condições, assim como as relações entre eles. Fornecendo ainda a capacidade de modelação, com o sincronismo de processos, concorrência, conflitos e partilha de recursos [Peterson, 77]. Através das RdP consegue-se capturar a dinâmica dos sistemas e eventos discretos tornando-se assim particularmente úteis para simulação [Murata, 89] [CSD, 09].

### 3.2.3- Estrutura das Redes de Petri

Quanto à representação gráfica das RdP esta é feita através de um grafo bipartido com dois tipos de nós: lugares e transições. Estes nós são interligados através de arcos havendo assim três elementos constituintes numa RdP (embora haja autores que considerem quatro, pois contam com as marcas [Moen, 03]) [Murata, 89].

Estes três elementos são então:



Figura 3.15 - Lugar

**Lugares** – estão associados a uma condição ou a um estado do sistema, podendo ainda estar associados a um subsistema. É através destes que se modela a componente estática, podendo assim observar-se o estado do sistema. Contêm marcas (*tokens*) que determinam a dinâmica do sistema e são representadas como pontos. Estas podem representar os recursos do sistema. A capacidade dos lugares pode ser fixa, ou não. Entre lugares surgem sempre arcos e transições forçosamente. Os lugares são representados por circunferências ou elipses como apresentado na Figura 3.15 [Pais, 04] [Barros, 96].





Figura 3.16 - Transição

**Transições** – são responsáveis pela mudança/evolução de estado, destruindo e/ou criando marcas. Como estão sempre entre lugares é através da acção destas (denominado disparo) que um lugar altera a sua marcação. São componentes activos correspondendo a eventos que vão acontecendo dentro do sistema. São representadas por segmentos de recta, rectângulos ou barras como na Figura 3.16 [Barros, 96].



Figura 3.17 - Arco

**Arcos** – fazem a ligação entre os lugares e as transições, indicando os lugares sobre os quais a transição actua. A sua função é transportar marcas. A estes estão associados pesos, que indicam o número de marcas que transportam. Podemos considerar dois tipos de arcos, arcos de saída e arcos de entrada. O peso de um arco de entrada representa a quantidade de marcas necessária no lugar de origem de modo a que a transição a que está ligada possa disparar. O peso do arco de saída indica o número de marcas que serão criadas quando ocorrer o disparo no lugar destino. Nunca podem interligar dois elementos do mesmo tipo. Representam-se através de setas como mostrado na Figura 3.17. [Conceição, 04] [Barros, 96].

Assim, com estes elementos podemos criar uma rede como a que é mostrada na Figura 3.18. Esta RdP é um exemplo simples de um modelo de um protocolo de comunicação. Quanto ao seu funcionamento, será explicado mais á frente quando for abordado o disparo das transições. A RdP apresentada é designada como uma RdP marcada, isto porque contém marcas nos lugares.

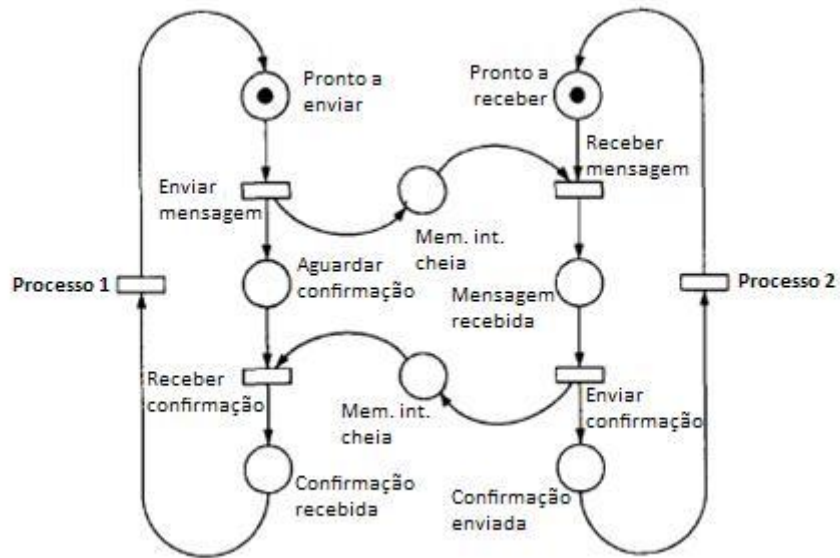
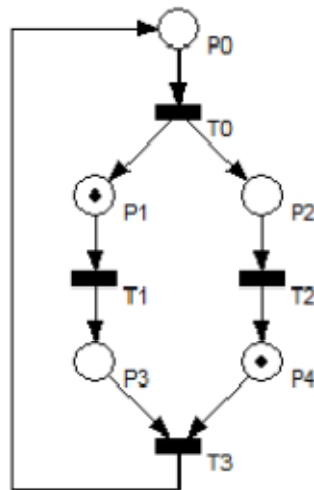


Figura 3.18 – Exemplo de uma Rede de Petri, adaptado de [Murata, 89]

Quanto à estrutura de uma RdP há autores que consideram que esta é composta por quatro elementos [Peterson, 77], enquanto outros consideram cinco [Murata, 89] contando com a marcação inicial. Está relacionado com o facto de ser uma RdP marcada ou não marcada. Assim segundo a definição em [Reisig, 85] uma RdP é representada por  $PN = (P, T, I, O, M_0)$  em que PN designa *Petri Net* (RdP), P de *Place* (lugar), T de *Transition* (transição), I de *input* (entrada), O de *output* (saída), M de *Marking* (marcação) [Reis, 08].

- $P = \{p_1, p_2, \dots, p_n\}$  – conjunto dos lugares,  $n \geq 0$
- $T = \{t_1, t_2, \dots, t_m\}$  – conjunto dos transições,  $m \geq 0$
- $I = P \times T$  – conjunto de arcos de entrada nas transições
- $O = T \times P$  – conjunto de arcos de saída nas transições
- $M_0 = (m_1, m_2, \dots, m_n)$  – marcação inicial

Deste modo  $n, m \in \mathbb{N}$  em que  $n$  representa o número de lugares, e  $m$  representa o número de transições da RdP (não confundir com  $m_n$  que diz respeito à marcação).



**Figura 3.19** – Rede de Petri marcada, extraído de [Reis, 08]

Vejamos um exemplo de como se pode definir uma RdP. Na Figura 3.19 mostra-se uma RdP marcada, que pode ser representada da seguinte maneira:

- $P = \{P_0, P_1, P_2, P_3, P_4\}$
- $T = \{T_0, T_1, T_2\}$
- $I = \{(P_0, T_0), (P_1, T_1), (P_2, T_2), (P_3, T_3), (P_4, T_3)\}$
- $O = \{(T_0, P_1), (T_0, P_2), (T_1, P_3), (T_2, P_4), (T_3, P_0)\}$
- $M_0 = (0, 1, 0, 0, 1)$

### 3.2.4- Disparo das transições

São as transições que permitem que o modelo evolua. Como referido anteriormente, estas têm a capacidade de criar ou destruir marcas, consoante o peso dos arcos associados. Este mecanismo de disparo acontece quando estas estão habilitadas. As transições ficam habilitadas quando todos os lugares de entrada contêm uma ou mais marcas, ou então o número de marcas igual ou superior ao peso do arco que liga esse lugar à transição [Barros, 96].

Vejamos uns exemplos do disparo de algumas transições:

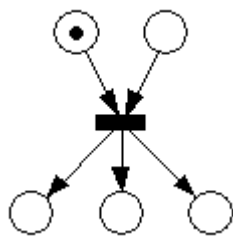


Figura 3.20 – Transição desabilitada

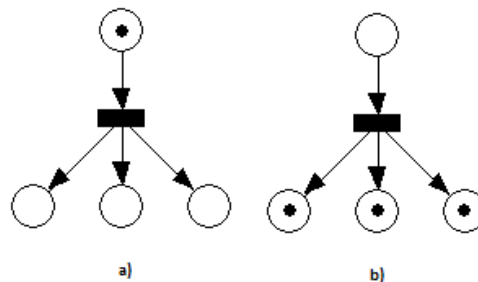


Figura 3.21 – Disparo de uma transição. a) situação inicial. b) situação após o disparo

Na Figura 3.20 surge uma transição que está desabilitada, pois nem todos os lugares de entrada têm marcas. Já na Figura 3.21 surge uma transição habilitada em a) e que após o disparo criou/adicionou três marcas nos lugares de saída. Já na Figura 3.22 é apresentado um exemplo que os arcos já têm pesos diferentes. Podemos ver que quando a transição dispara nos lugares de entrada são destruídas as marcas correspondentes ao peso de cada arco, e nos lugares de saída são criadas novamente conforme os pesos dos arcos.

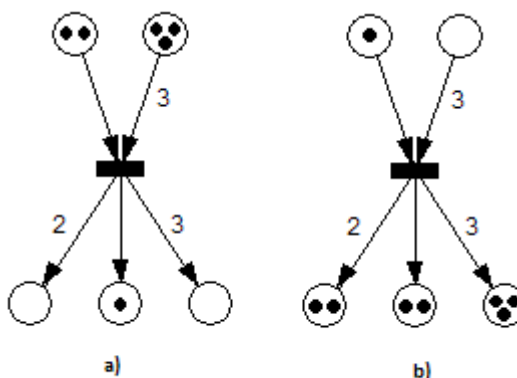


Figura 3.22 – Disparo de uma transição (arcos com pesos).  
a) situação inicial. b) situação após o disparo

Após explicado o mecanismo de disparo, explica-se o funcionamento da RdP apresentada na Figura 3.18. Inicialmente o processo 1 pretende comunicar com o processo 2. Depois de enviada a mensagem o processo 1 fica à espera de confirmação (*acknowledge*) e a memória intermédia (*buffer*) que é partilhada pelos dois fica cheia. Deste modo o lado do processo 2 pode receber a mensagem, e em seguida enviar o *acknowledge*. Como o lado processo 1 já estava à espera deste, recebe o *acknowledge* e executa novamente o processo 1. Logo em seguida a RdP volta ao estado inicial com o processo 1 a querer enviar e o processo 2 a querer receber uma mensagem.

### 3.2.5- Modelação com Rede de Petri

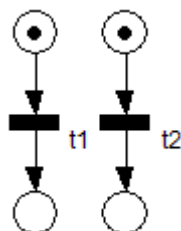
Ao modelar um sistema através de uma RdP, estamos a criar uma interpretação da rede (não confundir com “RdP interpretada”). “É essa interpretação ou significação que efectua a ligação do modelo abstracto que qualquer RdP representa, com o sistema concreto que se pretende modelar” [Barros, 96].

As RdP têm a capacidade de modelar e visualizar vários conceitos [Lino, 03]:

- Paralelismo
- Concorrência
- Localidade
- Conflito
- Partilha de recursos
- Sincronização
- Memorização
- Limitação de recursos
- Teste à marcação

“Num modelo de fluxo de dados (*dataflow*) os operadores são activados pela chegada dos operandos. Nas RdP os operandos representam-se através das marcas nos lugares e os operadores encontram-se associados a transições”[Barros, 96].

Na modelação com RdP são típicas as seguintes situações:



**Paralelismo/Concorrência** – mais que um processo são executados em simultâneo (Figura 3.23).

Figura 3.23 – Paralelismo

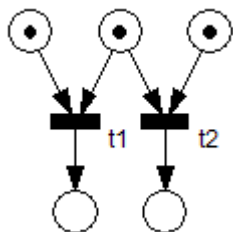


Figura 3.24 - Conflito

**Conflito** – tanto a transição  $t_1$  como  $t_2$  da Figura 3.24 estão habilitadas ocorrendo assim um conflito sobre qual dispara primeiro. Ao disparar uma, a outra deixa de estar habilitada.

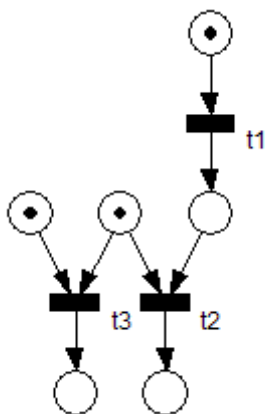


Figura 3.25 - Confusão

**Confusão** – é a mistura de concorrência e conflito. Na Figura 3.25 se a transição  $t_1$  disparar primeiro que a  $t_3$  resulta numa situação de conflito.

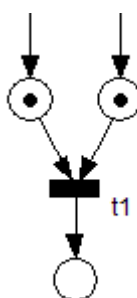


Figura 3.26 - Sincronização

**Sincronização** – na Figura 3.26 a transição  $t_1$  só fica habilitada quando os dois lugares estão marcados. Deste modo sincroniza-se dois processos que tenham tempos de execução diferentes, mas que a partir de  $t_1$  tenham que forçosamente ter terminado.

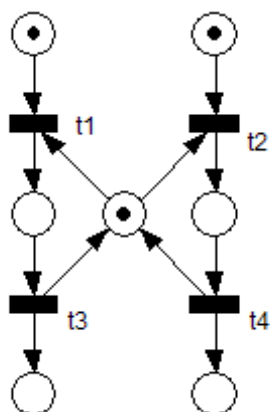
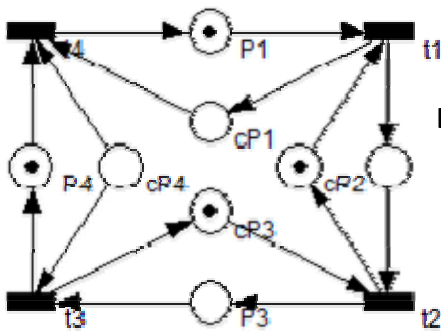


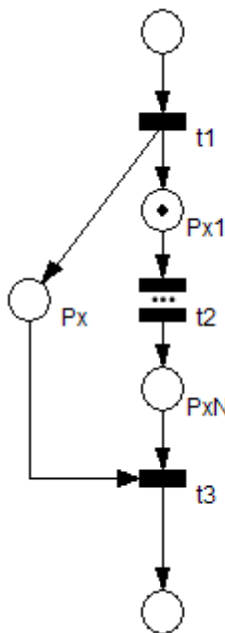
Figura 3.27 – Gestão de recursos

**Gestão de recursos** – na Figura 3.27 surge uma RdP que permite a gestão de um recurso crítico. Tanto  $t_1$  como  $t_2$  estão habilitadas, mas ambas necessitam do mesmo recurso, logo apenas uma pode disparar. Quando uma dispara, utiliza o recurso e no fim as transições  $t_3$  e  $t_4$  voltam a repor o recurso disponível.



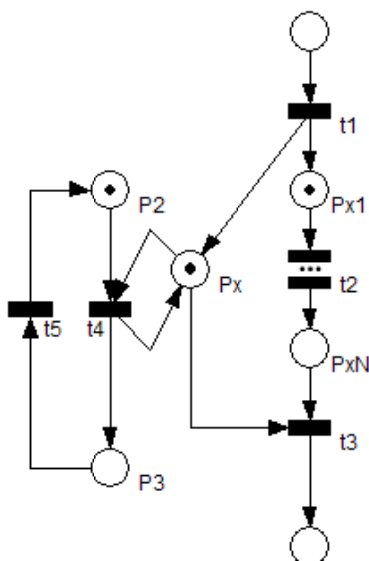
**Lugares complementares** – na Figura 3.28 apresenta-se uma RdP com lugares complementares. Os lugares complementares têm sempre o complemento da marcação do lugar a que estão associados.

Figura 3.28 – Lugares complementares



**Memória** – na Figura 3.29 apresenta-se uma RdP com capacidade de memória. O lugar  $P_x$  sinaliza que o processo X está a ocorrer. Como o processo X é complexo, necessita de vários estados, de  $P_{x1}$  a  $P_{xN}$  para o concluir. Enquanto o processo vai percorrendo os vários lugares, o  $P_x$  está sempre marcado. Uma aplicação típica, seria se fosse necessário habilitar uma transição, enquanto este processo não terminasse. Para isso teria que ser feito com um arco de teste.

Figura 3.29 - Memória



**Arco de teste** – na Figura 3.30 é apresentado o exemplo anterior mas agora com a aplicação de um arco de teste. O arco de teste nunca consome a marca do lugar, serve apenas para habilitar a transição. Assim enquanto o processo X ocorre,  $P_x$  está marcado, e  $P_2$  e  $P_3$  vão ocorrendo. Quando  $t_4$  disparar,  $P_3$  nunca mais é atingido, pois  $P_x$  deixou de habilitar  $t_5$  [Marranghello, 05] [CSD, 09].

Figura 3.30 – Arco de teste

### 3.2.6- Classes de Redes de Petri

Como referido anteriormente nesta dissertação, com a evolução das RdPs foram surgindo variações ao modelo inicial, chamadas classes. Estas classes vão surgindo à medida que as RdP são utilizadas em novas áreas. Como por exemplo existem as RdP coloridas, RdP Reactivas (RdP-R), RdP Reactivas e Hierárquicas (RdP-RH), RdP de controlo interpretado por cor CCIPN (*Coloured Control Interpreted Petri Nets*), entre outras [Barros, 96] [Wegrzyn et al, 97] [Gomes, 97] [Jensen, 97].

Para a classificação das RdP existem dois critérios. Um critério tem em conta as dependências em relação a características físicas e outras, alterando as regras de disparo das transições e conduzindo a uma classificação em dois grandes grupos: RdP autónomas e RdP não autónomas. Outro critério classifica o tipo de marcas. Se estas são indistintas entre si, classifica-se como RdP de Baixo-Nível, enquanto que as de Alto-Nível têm uma estrutura de dados associada [Barros, 96 ].

Das RdPs Baixo-Nível as mais utilizadas são as RdP Lugar-Transição, e das de Alto-Nível são as RdP Coloridas.

Na Tabela 3.1 são apresentadas algumas classes e as suas classificações.

<b>RdP</b>	<b>Não-Autónomas</b> estão directamente ligadas à modelação de sistemas físicos	<b>Autónomas</b> são aquelas a que não se associa nenhuma interpretação
<b>Baixo Nível</b> são caracterizadas por lugares marcados em que as marcas são indistintas entre si	<b>Interpretadas Sincronizadas Temporizadas</b>	<b>Condição-evento Elementares Lugar-transição</b>
<b>Alto Nível</b> são caracterizadas por lugares que podem conter marcas com uma estrutura de dados associada	<b>Reactivas</b>	<b>Coloridas</b>

**Tabela 3.1** - Classes das RdP [Lino, 03]



## Classe IOTP

As redes IOPT (Input-Output Place-Transition) são baseadas nas redes Lugar/Transição (de baixo nível como apresentado na Tabela 3.1). As redes IOPT foram formalmente apresentadas em [Gomes et al, 07] embora este conceito já existisse algum tempo antes. As redes IOPT estendem a classe de RdP Lugar-Transição na medida em que permitem interações entre a rede/modelo com o meio externo. Assim, o meio externo contém entradas e saídas de eventos e sinais. Deste modo, permitem o uso de sinais e eventos externos de entrada bem como eventos de saída associados às transições e ainda a especificação de sinais externos de saída associados a lugares. Quanto à classificação, estas são redes não-autónomas.

Em [Gomes et al, 07] são apresentadas as definições, propriedades e características desta nova classe. São ainda apresentadas as ferramentas de desenvolvimento para as redes IOPT. Estas redes permitem a sua representação em PNML, o que facilita o desenvolvimento de ferramentas de apoio a esta classe.

### 3.2.7- PNML

O PNML (Petri Net Markup Language) é um formato de representação de RdP. Este é baseado no XML (Extensible Markup Language) que especifica diferentes campos/atributos através de etiquetas (*tags*) tal como no HTML. O PNML surge com os trabalhos apresentados em [Jünger et.al, 00] e [Weber-Kindler, 02], como a norma de representação para as RdP. Através de um formato único e universal é possível uma maior portabilidade das RdP entre ferramentas e projectos. O PNML apresenta-se com os seguintes princípios:

**Legibilidade** – este formato deve ser legível e editável por humanos através de qualquer editor de texto;

**Universalidade** – o formato não deve excluir nenhuma versão das RdP. Deverá ser possível a representação de qualquer RdP com qualquer tipo de extensão;

**Mutabilidade** – este formato deve permitir representar toda a informação possível de uma RdP, até mesmo se o tipo de RdP for desconhecido. Deste modo o formato deve representar os princípios comuns das RdP [Weber-Kindler, 02].

Através da utilização do XML garante-se a legibilidade, enquanto que a universalidade é implementada através da definição das etiquetas, valores e combinações de valores autorizadas num arquivo chamado Petri Net Type Definition (PNTD). Já a mutabilidade é garantida através de convenções que definem conjuntos de etiquetas permitidas.

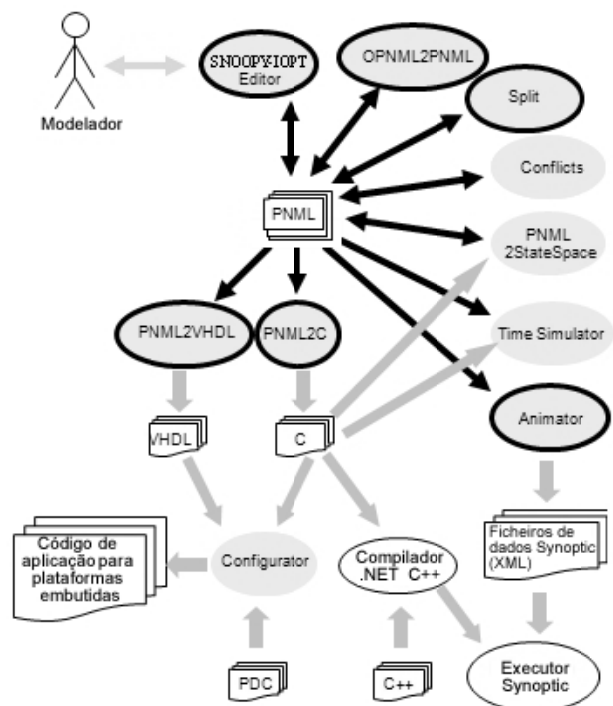
Um factor que foi tido em conta foi o facto dos sistemas reais serem demasiadamente grandes para que o seu desenho caiba numa página apenas. Assim, as ferramentas de desenho têm mecanismos para edição de sistemas grandes através de dois mecanismos de estruturação, a referência a páginas e através de módulos [Weber-Kindler, 02].

Na referência de páginas ou PNML estruturado (*Structured PNML*) cada rede é uma página, e cada página pode referenciar outras, sendo assim possível fazer uso de redes que se encontram noutras páginas. Utilizando PNML modular (*Modular PNML*) cada rede é um módulo. Assim, cada módulo pode fazer uso dos módulos existentes ligando-os apenas através de nós de importação/exportação.

O PNML estrutural fornece uma forma simples de partir um modelo em vários submodelos através de referências. O PNML modular permite a criação de várias instâncias independentes do modelo inicial, sendo muito útil uma vez que cada módulo pode ser utilizado em vários contextos distintos em simultâneo [Barros-Gomes, 04].

Em [Barros-Gomes, 04] apresentam-se duas operações genéricas para a especificação e modificação do modelo, chamadas adição à rede (*net addition*) e subtracção à rede (*net subtraction*). Estas operações são complementares à definição do modelo das RdP através do PNML. Deste modo, é proposto o PNML operacional (*Operational PNML*) que possibilita a definição de modelos de RdP como sequências complexas e arbitrárias de operações a qualquer instância da rede. Essas operações incluem a adição à rede que é uma generalização de fusão/acrescentamento de nós, mas ainda a operação inversa a subtracção à rede.

O PNML foi utilizado no trabalho desenvolvido nesta dissertação como elo de ligação entre o editor Snoopy IOPT [Gomes et al, 07] e o conversor de código para a linguagem C (PNML2C). Estas ferramentas foram e continuam a ser desenvolvidas pelo projecto FORDESIGN que tem como objectivo a criação de diversas ferramentas de apoio à modelação de sistemas por RdP, fazendo uso do PNML como linguagem neutra intermédia. É mostrada a relação entre estas ferramentas na Figura 3.31 [FORDESIGN, 07].



**Figura 3.31** – Ferramentas ForDesign, adaptado de [ForDesign, 07]

### 3.3- UML

O UML fornece notações e técnicas de modelação que têm vindo evoluindo nas últimas décadas. Tem sido formalmente desenvolvido desde 1994. Actualmente é sem dúvida a norma para a modelação de *software* orientado ao objecto mais utilizada. Esta tem sido adoptada por várias companhias e existem disponíveis diversas ferramentas, quer profissionais quer académicas, que lhe dão suporte.

O UML fornece ao desenvolvimento de um sistema a especificação e documentação de modo a garantir escalabilidade, segurança e robustez. As técnicas de modelação UML

facilitam a criação de desenhos modelares, traduzindo-se em módulos isolados que vão formando uma biblioteca de módulos, auxiliando na reutilização de código/trabalho previamente desenvolvido.

A arquitectura escrita no UML é baseada no Meta-Object Facility (MOF). O MOF define formatos normalizados para que os elementos principais de um modelo possam ser guardados num repositório e partilhados entre ferramentas de modelação e linguagens. É através do XMI (XML Metadata Interchange) que são garantidos os mecanismos de troca e partilha. Graças a esta uniformização, a escolha de ferramentas a utilizar reside inteiramente nas pessoas que desenvolvem o modelo e o concretizam.

Os modelos UML podem ser tão precisos que permitem a geração de código. Existem ainda soluções que verificam a precisão do modelo. Assim, quando se utiliza o UML com ferramentas deste género, é possível testar o modelo ainda antes da produção de código manualmente. Um modelo UML pode ser implementado em múltiplas plataformas que tiram partido de diferentes linguagens. Deste modo, o UML apresenta-se com um grande grau de independência entre o modelo e a plataforma na qual é implementado [Pender, 03].

Como referido, o UML fornece diversas notações e técnicas de modelação, como por exemplo, diagrama de casos de uso, diagrama de classes, diagrama de componentes, diagrama de pacotes, diagrama de actividade, diagrama de estado, diagrama de sequência, diagrama de interacções, entre outros.

Em seguida abordam-se as técnicas de modelação/notações que foram utilizadas neste trabalho, nomeadamente os diagramas de casos de uso na identificação dos usos do sistema na óptica do utilizador, e os diagramas de sequência na representação da interacção e troca de mensagens entre os elementos da rede.

### **3.3.1- Diagrama de Casos de Uso**

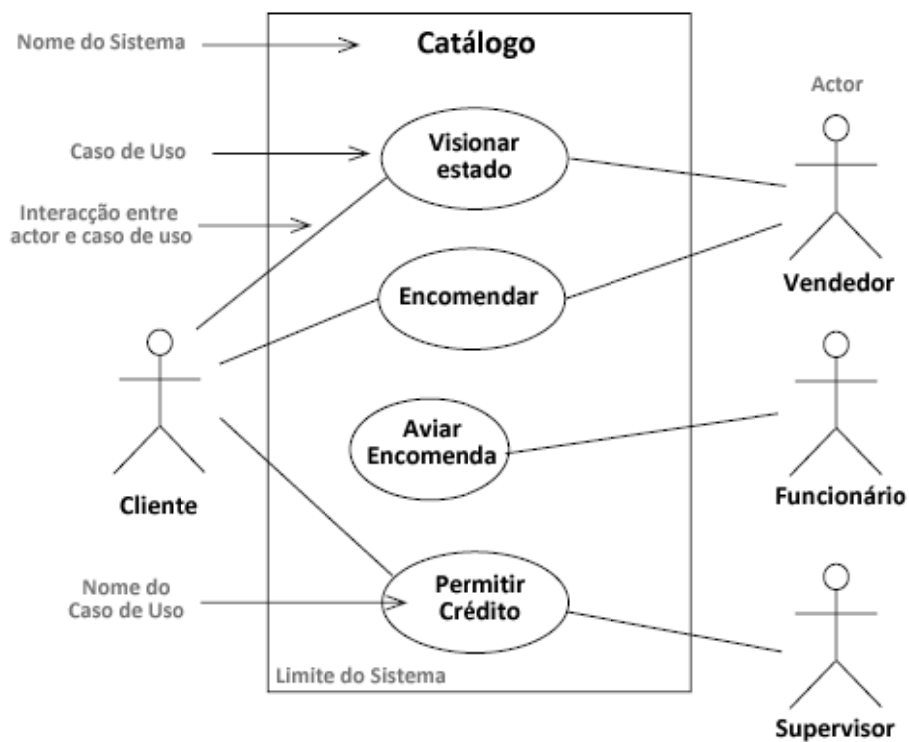
Os diagramas de casos de uso capturam o comportamento de um sistema, subsistema ou classe do ponto de vista de um utilizador externo, denominado actor. Estes demarcam as funcionalidades do sistema em função das interacções com os actores. Os mecanismos desencadeados são chamados casos de uso, e são estes que

se encontram representados no diagrama. Os casos de uso descrevem a sequência de acções entre o sistema e os respectivos actores.

O objectivo dos casos de uso é fornecer uma visão externa da relação entre o sistema e o mundo real. É um diagrama plano (não hierárquico) que confere a visão encapsulada da “caixa-negra” (*black-box*) ao sistema.

Como actor entende-se qualquer humano, ou qualquer sistema/processo que não o do modelo em questão, e que possa tomar iniciativa, desencadeando um conjunto de acções no sistema a modelar [Rumbaugh et al, 99] [Pender, 03].

Na Figura 3.32 apresenta-se um exemplo de diagrama de casos de uso, que representa o processo de compra de um produto através de um catálogo.



**Figura 3.32** – Exemplo de diagrama de casos de uso, adaptado de [Rumbaugh et al, 99]

Neste exemplo o cliente tem ao seu dispor a possibilidade (casos de uso) de efectuar encomendas, ver o estado das encomendas, e requisitar crédito. Por sua vez o resultado destes casos de usos pode estar dependente de outros actores, como o caso de requisitar crédito.

### 3.3.2- Diagramas de Sequência

O diagrama de sequência representa as interações num gráfico a duas dimensões. Na vertical o eixo do tempo, e na horizontal está representado o papel dos vários elementos, como a sequência de acções. Estas interações são representadas pela troca de mensagens entre os vários elementos.

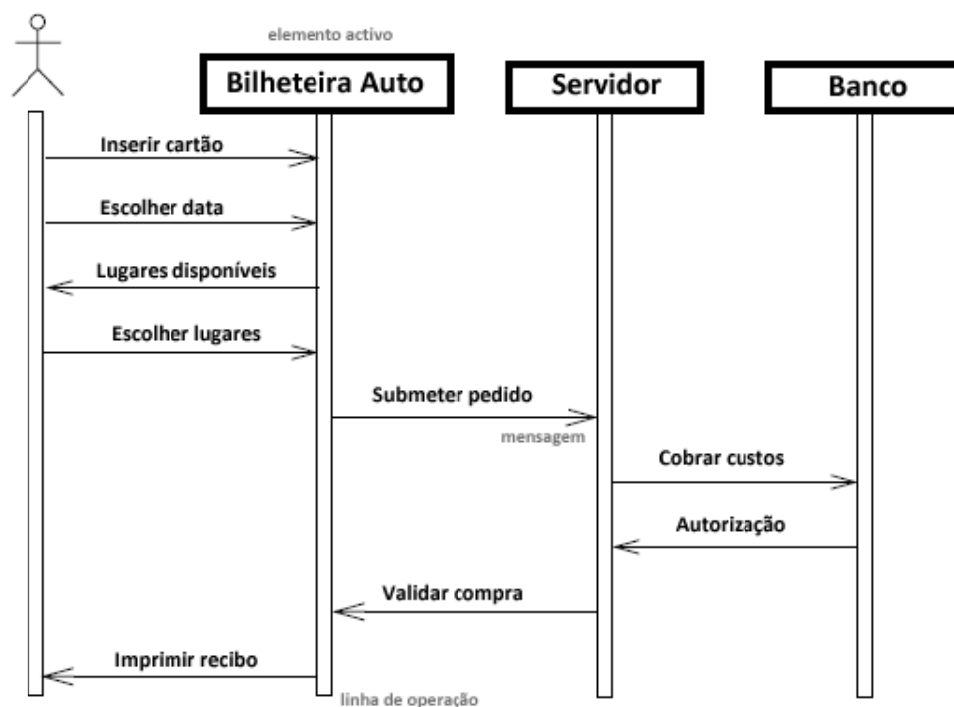


Figura 3.33 - Exemplo de diagrama de sequência, adaptado de [Rumbaugh et al, 99]

Na Figura 3.33 mostra-se um exemplo de um diagrama de sequência que representa o processo de compra de um bilhete numa bilheteira automática. De início o cliente introduz o cartão, em seguida escolhe a data que pretende. Depois, a bilheteira apresenta os lugares que se encontram disponíveis nessa data. O cliente escolhe então os lugares formalizando o pedido. Este pedido vai ser enviado a um servidor, que por sua vez vai pedir ao banco que valide o pagamento. No fim de autorizado o pagamento, o banco envia a validação ao servidor, que indica à bilheteira que o pedido foi efectuado e esta imprime o pedido.

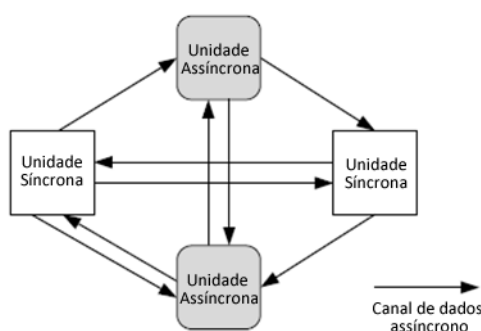
## 4- Sistemas GALS

Neste capítulo aborda-se a temática relativa aos sistemas *Globally Asynchronous Locally Synchronous* (GALS). Nesta caracterização apresenta-se o conceito destes sistemas, as suas implementações, as suas vantagens e desvantagens e são ainda apresentados os resultados de vários testes de desempenho.

Os sistemas GALS fornecem um método seguro de interligar dois sistemas em domínios temporais diferentes (síncrono e assíncrono). Verifica-se que a aplicação destes sistemas aumenta a complexidade dos desenhos, e pode levar a uma perda de desempenho (*performance*), mas demonstra uma redução do consumo energético e uma grande flexibilidade para sistemas até antes impossíveis. A metodologia GALS é um tema que tem vindo a ser amplamente estudado e testado, pois oferece características e oportunidades únicas.

### 4.1- Introdução

A primeira abordagem a sistemas GALS foi feita em 1984 [Chapiro, 84] e pretende descrever como fazer a ligação entre módulos com funcionamentos distintos, síncronos e assíncronos. Na Figura 4.1 mostra-se o esquema de comunicação entre módulos.



**Figura 4.1** Comunicação assíncrona entre módulos, adaptado de [Muttersbach et al, 00]

As linguagens síncronas têm sido amplamente utilizadas para o desenvolvimento quer de *software* quer de *hardware* [Benveniste et al, 02]. Todos estes sistemas baseiam-se no facto de serem síncronos a um sinal de relógio (*clock*). Este simples princípio pode ser bastante limitativo. Num sistema distribuído de tempo-real ter um sinal de relógio

global a todos os módulos pode ser demasiado dispendioso, ou mesmo impraticável. Num sistema à escala nanométrica, o atraso de propagação do sinal de relógio ao longo do circuito integrado (Integrated Circuit - IC) pode-se traduzir num grande obstáculo [Mousavi et al, 04].

Devido à redução do tamanho das tecnologias de implementação e ao aumento da dimensão dos desenhos é cada vez mais difícil distribuir um sinal global de relógio. Os sistemas assíncronos não têm esta dependência. Assim sendo é de prever uma mudança de paradigma de síncrono para assíncrono, por parte da indústria dos processadores num futuro “próximo”. É neste contexto que o estudo de sistemas GALS se torna relevante [Iyer-Marculescu, 02].

Inicialmente, o esquema proposto por M. D. Chapiro [Chapiro, 84] baseava-se em alguns princípios impraticáveis em desenhos modernos, mas foi o seu trabalho que foi a base de vários trabalhos recentes [Muttersbach et al, 00].

Em 1996 foi criado um sinal de relógio capaz de se “pausar”, *Pausable Clock Control* (PCC) [Yun-Donohue, 96], o que permitia a transferência de dados entre módulos com diferentes sinais de relógio. Assim conseguiu-se estender o sinal de relógio e fazer uso de *handshake* entre módulos. O uso do PCC estava limitado a sistemas reduzidos [Muttersbach et al, 00]. Mais tarde foi desenvolvido o conceito de invólucro/encapsulamento assíncrono (*asynchronous wrapper* -AW) [Yun-Donohue, 96] que permitiu mais flexibilidade na transferência de dados que o PCC [Muttersbach et al, 00].

As implementações actuais fazem uso destes conceitos com algumas alterações.

## **4.2- Conceitos dos Sistemas GALS**

Existem várias concretizações de sistemas GALS. Muitos trabalhos utilizam o conceito de AW [Muttersbach et al, 00], [Royal-Cheung, 03], [Najibi et al, 05], [Krstic et al, 06] outros propõem algumas alterações como os *synchro-tokens* [Heath et al, 05].

Neste capítulo vão ser descritos mais em pormenor os conceitos apresentados pelas referências [Muttersbach et al, 00] e [Najibi et al, 05].



O grande problema com os módulos síncronos embutidos passarem para um ambiente assíncrono, é conseguir a transformação entre dois paradigmas distintos de sincronização [Muttersbach et al, 00].

As transições de sinal no domínio assíncrono não são afectadas por nenhuma grelha temporal, portanto o comportamento de metaestabilidade nos biestáveis (*flip-flops*) dos módulos localmente síncronos (LS) é provável.

Metaestabilidade é um problema que ocorre quando um dado ou uma entrada de controlo é mudado no momento do pulso do sinal de relógio, originando um comportamento imprevisível na saída. Quando acontece, leva muito mais tempo que o seu normal para estabilizar no seu estado correcto. Por vezes pode ainda oscilar uma série de vezes antes de se estabilizar, podendo levar o sistema a um estado de bloqueio (*deadlock*), ou mesmo a avarias no *hardware* [Stein, 03].

Quando as alterações das entradas influenciam o comportamento dum sistema a ordem em que os eventos ocorrem determina o resultado. Quanto menor for o intervalo de tempo entre os eventos, mais tempo levará a determinar o resultado. Quando dois eventos ocorrem muito próximos o processo de decisão leva muito mais tempo do que o normal e ocorre um problema de sincronização [Cummings, 01].

#### **4.2.1- Encapsulamento Assíncrono**

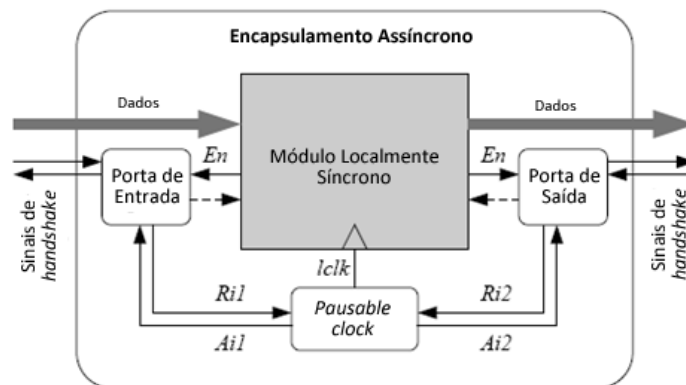
Para que se consiga comunicar entre domínios diferentes são utilizados os AWs. Rodeando os módulos síncronos com AWs consegue-se ter externamente uma interface completamente assíncrona. Para atingir este fim são utilizados sinais de *handshake* cada vez que são trocados dados, quer de entrada, quer de saída.

O *handshake* é feito por um par de *Request*, *Acknowledge*, que é executado por um protocolo de quatro fases. Inicialmente utilizou-se apenas *push channel*, que significa que a linha de *Request* é iniciada por quem transmite. Facilmente se estendeu o conceito para *pull channels* [Peeters, 96] [Muttersbach et al, 00].

Para que os AWs possam encapsular um módulo síncrono, têm que ser capazes de fornecer um sinal de relógio aos LS. Verifica-se assim que este facto é uma grande

vantagem pois permite-nos escolher diferentes frequências conforme as nossas necessidades em cada módulo. Por outro lado como este sinal de relógio pode ser controlado, é possível estendê-lo/esticá-lo de modo a que se uma transmissão de dados ocorrer muito próximo de uma transição do sinal de relógio possamos estender este, evitando assim o problema de metaestabilidade [Muttersbach et al, 00].

Na Figura 4.2 pode-se observar um módulo funcional de um sistema GALS com apenas uma porta de entrada e saída, mas é possível usar um número de portas qualquer.



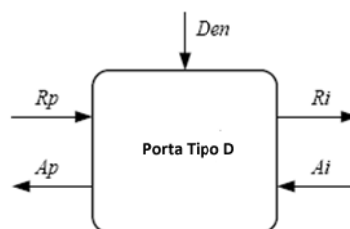
**Figura 4.2** - Módulo funcional num sistema GALS, adaptado de [Muttersbach et al, 00]

Outra vantagem da utilização de AWs é o facto de fornecer uma organização interna modular, permitindo assim a criação de vários modelos de modo a gerar uma biblioteca de módulos independentes, todos funcionais entre si.

#### 4.2.2- Controlador de Portas

Para circuitos globalmente assíncronos é necessário usar-se um esquema de comunicação assíncrona. Neste tipo de comunicações podem-se usar protocolos de *handshake* de duas ou quatro fases [Sparso- Furber, 02]. As linhas de dados estão agrupadas com uma linha de *request* que sinaliza a validação dos dados no canal.

Vários trabalhos apresentam dois tipos de controladores de portas: portas *demand* e portas *pull* [Muttersbach et al, 00] [Bormann- Cheung, 97] [Sparso- Furber, 02].

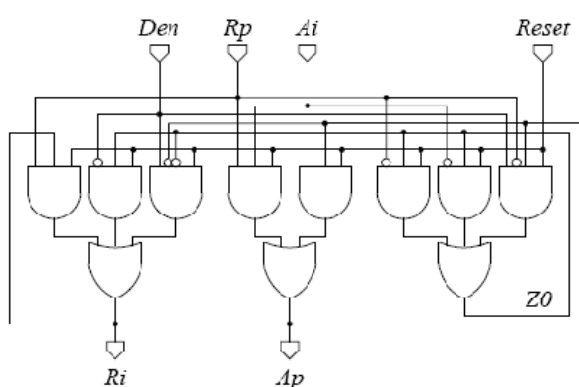


**Figura 4.3** - Esquema de uma Porta tipo-D [Muttersbach et al, 00]

Nas portas do tipo *demand* (tipo D) é assumido que os dados a ser transmitidos são requeridos imediatamente após comunicação, então neste tipo de portas o sinal de relógio do módulo LS deverá ser parado imediatamente, e reactivado quando a comunicação terminar [Najibi et al, 05]. O esquema destas portas apresenta-se na Figura 4.3.

Imediatamente após activação do evento Den, é feito um pedido para estender o sinal de relógio  $Ri+$  que é confirmado por um  $Ai+$ . Quando o sinal de relógio é garantido que permanece pausado, o ciclo é processado e no fim o sinal de relógio continua de novo.

Este tipo de porta pode ser conseguido, por exemplo, através da implementação apresentada na Figura 4.4.



**Figura 4.4** - Esquemático de um controlador D-type [Muttersbach et al, 00]

As portas do tipo *poll* (tipo P) não suspendem o clock imediatamente, e assim o circuito LS tem a oportunidade de continuar as suas operações e tarefas. Este tipo de porta foi desenhado para melhorar os desempenhos, mas a sua eficácia está altamente dependente da natureza das tarefas e do circuito do LS. Por isso, este tipo não é amplamente utilizado [Najibi et al, 05]. As portas tipo P têm necessidade de um sinal extra  $Ti$  que indica quando a transferência ocorreu [Muttersbach et al, 00].

#### 4.2.3- Gerador do sinal de relógio

A ideia do gerador de sinal de relógio é adiar o flanco ascendente até se concluir as comunicações de entrada/saída [Yun-Dooly, 99].

Se um controlador enviar um pedido *Ri+* para um “esticamento” do sinal de relógio o início do próximo pulso do sinal de relógio fica adiado até que o pedido persistir. A duração do pulso do sinal de relógio não deve ser influenciada, significando que só na fase *low* do sinal de relógio é que este poderá ser estendido [Muttersbach et al, 00].

A frequência do gerador do sinal de relógio deve ser ajustada precisamente usando uma cadeia de atraso [Taylor et al, 00]. O uso de uma cadeia de *buffers* pode limitar a frequência máxima do sinal de relógio [Najibi et al, 05]. O sinal de relógio é pausado na fonte, e é atrasado na sua distribuição ao longo do domínio síncrono.

Como referido anteriormente a implementação apresentada pela referência [Muttersbach et al, 00] faz uso do flanco descendente do clock para gerar *Den* e o flanco ascendente para outras partes do circuito. Isto é limitativo quando a lógica que gera *Den* é composta por circuitos que implementam grandes e complexas máquinas de estados [Najibi et al, 05].

Este método é usado porque causa pequenas modificações à máquina de estados inicial. A aplicação deste método resulta na redução da velocidade do circuito, pois o tempo é dividido ao meio entre as fases síncronas e assíncronas.

Na referência [Najibi et al, 05] é-nos sugerido o uso do flanco ascendente do sinal de relógio para gerar *Den*.

Trabalhos mais recentes propõem outras técnicas e metodologias para sinais de relógio aplicados a sistemas GALS como o caso da referência [Dasgupta- Yakovlev, 07].

#### 4.3- Resultados dos sistemas GALS

Existem inúmeros trabalhos que testam estes sistemas de modo a aferir os seus desempenhos [Najibi et al, 05], [Meinke et al, 98], [Iyer-Marculescu, 02]. Há ainda metodologias de teste para aplicações específicas [Meinke et al, 98], [Mousavi et al, 04].

De seguida apresentam-se os resultados de alguns destes trabalhos.

Em [Najibi et al, 05] é implementada uma arquitectura de modo a testar os sistemas GALS, usando uma FPGA Xilinx. Este trabalho apresenta um aumento de 11% no desempenho do circuito face a um circuito normal síncrono. Nesta implementação tira-se partido da possibilidade de ter diferentes módulos com frequências distintas.

Por outro lado a implementação através de sistemas GALS apresenta um acréscimo de 51% do tamanho e complexidade do circuito. Deste aumento de 51% aferiu-se que 22% são devido aos AWs, e os 29% restantes estão associados à replicação dos controladores das portas.

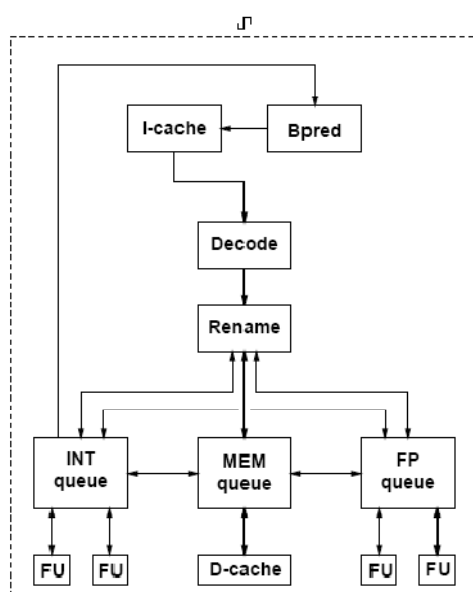
Relativamente ao consumo energético verificou-se que os circuitos GALS têm menores taxas de consumo. Neste teste o consumo foi inferior a cerca de 19% [Najibi et al, 05].

Na referência [Meinke et al, 98] é indicado que em termos de consumo, a utilização de sistemas GALS em arquitecturas VLSI, pode levar a uma redução do consumo energético de 30%.

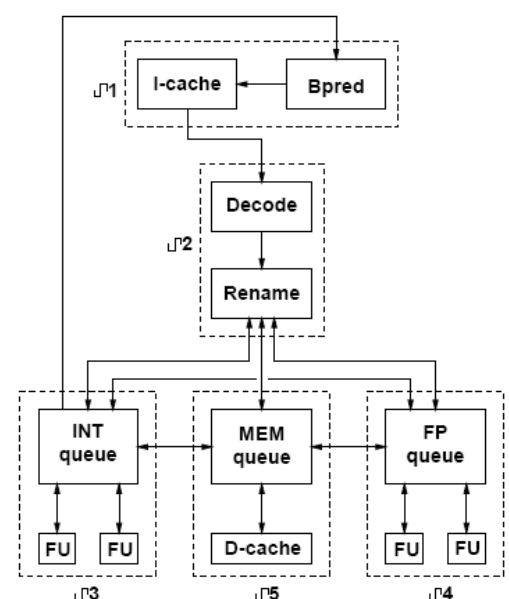
Já em [Iyer-Marculescu, 02] apresenta-se uma redução no desempenho do circuito usando o sistema GALS, isto devido aos *handshakes*.

Foram implementados dois processadores para os testes.

Através da Figura 4.6 e Figura 4.5 pode-se verificar as diferenças em termos do sinal de relógio do processador síncrono (base) e o processador GALS.



**Figura 4.5** - Processador síncrono (base),  
extraído de [Iyer-Marculescu, 02]



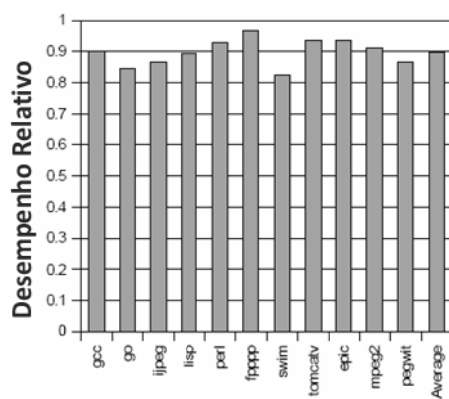
**Figura 4.6** - Processador GALS, extraído de  
[Iyer-Marculescu, 02]

Estes testes foram divididos em dois ensaios:

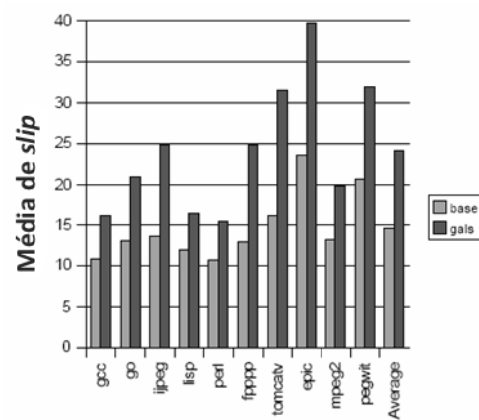
- O processador base *versus* o GALS com todos os módulos à mesma frequência e alimentação.
- O base *versus* o GALS com várias frequências e alimentações.

Nestes testes foram ainda executados diferentes algoritmos nos processadores para que o teste fosse mais representativo.

Os resultados para o primeiro ensaio são apresentados nas Figura 4.7 e Figura 4.8.



**Figura 4.7** - Desempenho do processador GALS relativo ao de base, de [Iyer-Marculescu, 02]

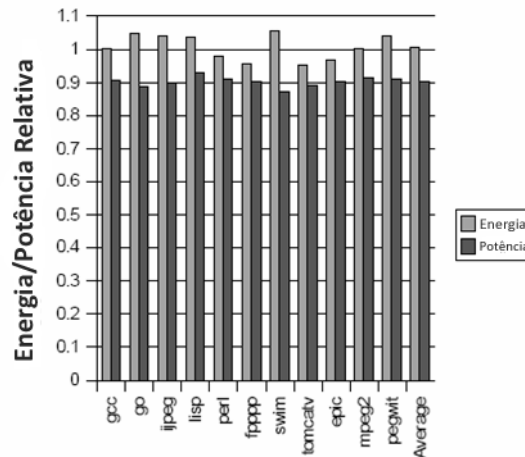


**Figura 4.8** - Média de *slip*, extraído de [Iyer-Marculescu, 02]

Através da Figura 4.7 verifica-se que há uma perda de desempenho ao utilizar-se o sistema GALS.

A Figura 4.8 mostra a média de *slip*. *Slip* é o tempo médio gasto por cada instrução desde a preparação (*fetch*) até ao *commit stage*. Verificou-se um aumento de cerca de 65% de *slip*.

Já em relação aos consumos, a Figura 4.9 apresenta o consumo total de energia relativo e a média de energia consumida do processador GALS em relação ao de base. Na maioria dos testes a eliminação do sinal de relógio global resulta em alguma poupança na dissipação de potência por ciclo.



**Figura 4.9** - Consumo do processador GALS relativo ao de base, extraído de [Iyer-Marculescu, 02]

No segundo ensaio testou-se o resultado de abrandar alguns domínios do sinal de relógio. Observa-se que há benefícios em termos de consumo mas algumas perdas na eficiência, cerca de 18%.

Assim conclui-se que o abrandamento não pode ser genérico, este depende da função do módulo e do algoritmo a executar pelo programa principal. Aplicando abrandamentos selectivos podemos maximizar o desempenho do sistema [Iyer-Marculescu, 02].

#### 4.4- Vantagens e desvantagens

Os sistemas GALS trazem algumas vantagens e desvantagens em relação aos sistemas síncronos tipicamente utilizados.

O uso de arquitecturas GALS abre um leque de oportunidades e vantagens no desenho de sistemas. Dum ponto de vista síncrono fornece uma completa liberdade das limitações temporais de cada módulo, e faz isso sem os receios da metaestabilidade [Muttersbach et al, 00].

A vantagem mais saliente é de facto possibilitar a interacção de sistemas com filosofias diferentes (síncronos e assíncronos). Mas não é apenas para esta situação que os sistemas GALS têm interesse, pois pode ser aplicado este conceito entre módulos síncronos mas que possam estar a utilizar sinais de relógio a diferentes frequências.

Deste modo é possível ter num sistema várias frequências em módulos diferentes consoante as necessidades de cada um.

Outra grande vantagem introduzida por este conceito é o facto de não ser necessário disponibilizar um sinal de relógio geral a todo o circuito, o que torna todo o desenho e desenvolvimento do hardware bastante mais simples e menos dispendioso.

Verifica-se ainda que existe uma redução nos atrasos devido à utilização de longos fios/pistas [Jia-Vermuri, 05]. Nas FPGA verificou-se que o uso de longos fios em caminhos críticos reduz o desempenho. Resultados experimentais demonstram que 60% a 80% dos atrasos relacionados com caminhos críticos são consumidos por interligações [Anderson-Najm, 03]. Esta situação tende a piorar consoante o crescimento natural dos chips, em particular das FPGA [Jia-Vermuri, 05].

Utilizando o conceito GALS têm-se vários módulos independentes entre si, cada um com as suas topologias a interagirem. Esta visão de todo o sistema permite um grande nível de modularização. Ao ter o sistema constituído por módulos independentes podemos fazer alterações ao sistema sem ter que o redesenhar, tornando o sistema muito melhor do ponto de vista de reutilização ou mesmo de reparação [Muttersbach et al, 00].

Outro dos resultados obtidos quando se estuda os sistemas GALS é o facto de estes apresentarem na generalidade uma diminuição nos consumos de energia [Iyer-Marculescu, 02], [Najibi et al, 05], [Meincke et al, 98], [Iyer-Marculescu, 02], [Hemani et al, 99]. Consequentemente este facto torna os sistemas GALS ainda mais importantes com as necessidades actuais de sistemas de baixo consumo e autonomias elevadas.

Porém a utilização destes sistemas resulta muitas vezes numa perda de desempenho do circuito devido aos mecanismos inerentes aos AWs. Por vezes esta perda de desempenho pode não ser crítica, consoante o propósito do nosso sistema, mas não deixa de ser um factor limitativo destes sistemas.

Outro problema associado ao uso destes sistemas é o facto de estas arquitecturas aumentarem bastante o tamanho e a complexidade do circuito, como referido em [Najibi et al, 05], onde houve um aumento de 51%. Este aumento da complexidade torna-se não tão grave graças à modularidade, uma vez que todo o sistema pode ser visto como módulos independentes.



#### 4.5- Utilizações de Sistemas GALS

Estes sistemas GALS estão a ser estudados devido às suas características especiais e prevê-se a sua implementação em diversas áreas.

Uma das grandes aplicações dos sistemas GALS é nas arquitecturas VLSI (*Very-large-scale integration*) ou mesmo Structured VLSI, que se baseiam na modularização de todo o sistema.

Dado o aumento do tamanho e velocidades de sinal de relógio mais elevadas, os futuros desenhos VLSI requerem uma mudança de paradigma do estilo habitual, globalmente síncrono. Mesmo a integração de vários IP (*Intellectual Property*) em sistemas complexos “*on chip*” têm a necessidade de múltiplas frequências de sinais de relógio. O paradigma GALS permite tal integração, pois permite módulos síncronos operarem assincronamente uns com os outros. Com sistemas GALS não só a frequência do sinal de relógio pode ser diferente, mas a alimentação de cada bloco pode ser “afinada” para encaixar nos requisitos do sistema.

Este paradigma é também atractivo para desenhos SoC (*System-on-Chip*) onde vários blocos são integrados num único *chip* [Niyogi-Marculescu, 05].

Outras aplicações dos sistemas GALS são os sistemas de baixo consumo (*low-power*), que actualmente têm tido um grande interesse. Uma das aplicações de baixo consumo mais usual é as WSN.

Como visto anteriormente na secção 2 desta dissertação, as WSN apenas têm como alimentação pequenas baterias/pilhas, e têm que executar tarefas como receber informação dos seus sensores, processar essa informação e comunicar de modo *wireless*.

Um grande obstáculo é conseguir aumentar o ciclo de vida da bateria. Para resolver este obstáculo é necessária uma abordagem radical ao consumo de energia dos nós das WSNs. É devido a estas dificuldades e restrições tão exigentes que estes sistemas são levados à metodologia de desenho de arquitecturas GALS [Fernández et al, 07].

## 4.6- Enquadramento

Embora os sistemas GALS não estejam directamente envolvidos no trabalho elaborado nesta dissertação, estão relacionados com os conceitos aqui aplicados.

Em [Brauer-Reisig, 06] explica-se que os trabalhos iniciais de C.A. Petri basearam-se muito no domínio do assíncronismo. São referidos trabalhos de Petri em que o conceito era constituído por módulos comunicando entre si, num ambiente assíncrono. Petri sempre se empenhou em demonstrar as vantagens duma modelação assíncrona e destes sistemas, em vez de sistemas síncronos. É referido que Petri defende que num ambiente assíncrono não faz sentido a modelação através de modelos sequenciais. E é neste contexto que as RdP, como as conhecemos hoje, surgiram. É assim que, de forma não explícita, as RdP estão relacionadas com a filosofia dos sistemas GALS. Parece ainda que o C.A. Petri foi um dos pioneiros de uma arquitectura assíncrona com módulos funcionais (1962, enquanto que o conceito GALS surge em 1984). A modelação por RdP parece a mais natural e indicada para sistemas com uma arquitectura GALS.

A utilização das RdP é extremamente útil e apropriada para a sincronização de processos. Estas permitem um sincronismo perfeito em sistemas distribuídos e ainda para desenhos de *hardware* como os sistemas GALS [Nielsen et al, 01] [Bystrov-Shang, 06].

Em [Yakovlev, 02] são apresentados alguns motivos pelos quais as RdP são apropriadas para a modelação destes sistemas, como a tradução entre as RdP e as linguagens HDL (Hardware Description Language), ou ainda as técnicas de composição e decomposição possíveis com as RdP.

As RdP são ainda apontadas como o formalismo a usar para a modelação de sistemas GALS em trabalhos futuros [Singh-Theobald, 03].

Desta forma surgiram alguns trabalhos que tiram partido das RdP em sistemas GALS.

Em [Dasgupta- Yakovlev, 06] são apresentados três esquemas de sinais de relógio para sistemas GALS através da modelação por RdP.

Já em [Stahl et al, 05] são usadas as RdP como técnica de modelação para os AWs das GALS e ainda na detecção de possíveis falhas. São ainda explicadas as vantagens na modelação por RdP.

Outro trabalho que utiliza as RdP como formalismo de modelação e teste, é o apresentado em [Krstić et al, 05].

É importante referir que grande parte destes trabalhos utiliza a ferramenta Petrify, que a partir de um modelo, chamado sistema de transição (*Transition System*), sintetiza uma RdP segura e fiel ao modelo inicial [Cortadella et al, 96].

Deste modo, verifica-se que as RdP começaram a ser utilizadas na investigação que está a ser desenvolvida na área das GALS, tornando-se numa das aplicações das RdP. Os sistemas GALS são uma área em que as RdP serão utilizadas, estudadas e aplicadas, uma vez que estes estão, nesta altura, a ter um incremento no interesse da comunidade científica.

Por sua vez o trabalho desenvolvido nesta dissertação, utiliza as RdP como formalismo de modelação, tal como os sistemas GALS têm vindo a utilizar. Estes sistemas estão também a ser utilizados na área das WSN, quer pelos ganhos de consumo quer pelas vantagens das suas propriedades.

Em [Krstić et al, 05] a aplicação da estrutura GALS na comunicação sem fios revelou-se vantajosa. Já em [Fernández et al, 07] é proposto um *bus* capaz de dar suporte a sistemas GALS num ambiente WSN, sobre uma plataforma SoC.

Mas não é só em termos de *hardware* que os sistemas GALS trazem vantagens às WSN. Como referido anteriormente na secção 2.3 desta dissertação, o TinyGALS é uma metodologia/arquitectura de desenho de *software*, baseado nos conceitos GALS, que tem sido utilizado nas WSN através do TinyOS [Cheong et al, 03].

## 5- Ferramentas Utilizadas

### 5.1- Ferramentas para as Redes de Petri

#### 5.1.1- Snoopy-IOPT

O Snoopy-IOPT é uma ferramenta de edição gráfica de redes de Petri para a classe IOPT com a capacidade de suportar especificações hierárquicas e modulares. Esta ferramenta foi desenvolvida no trabalho apresentado em [Nunes et al, 07].

Esta é um elemento fulcral no Projecto FORDESIGN, pois é através dela que se gera o PNML que será depois utilizado nas várias ferramentas deste projecto [FORDESIGN, 07], como se pode ver na Figura 3.31. A Figura 5.1 mostra o interface gráfico do Snoopy-IOPT.

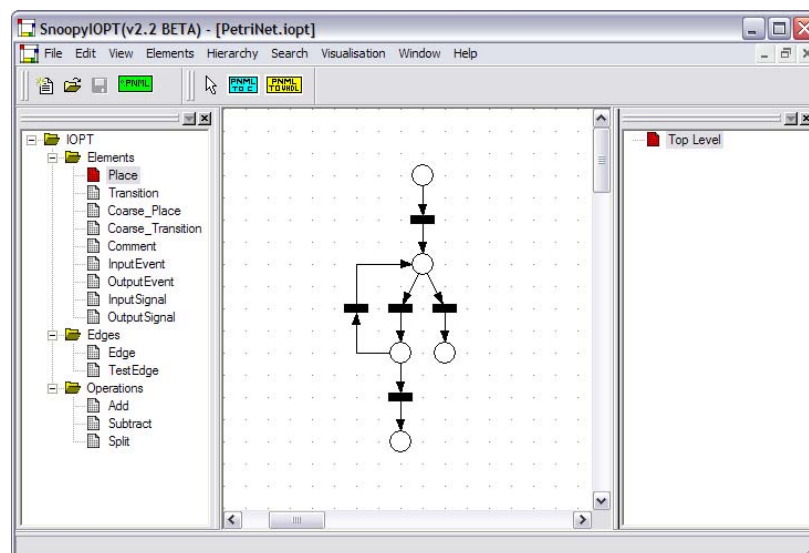
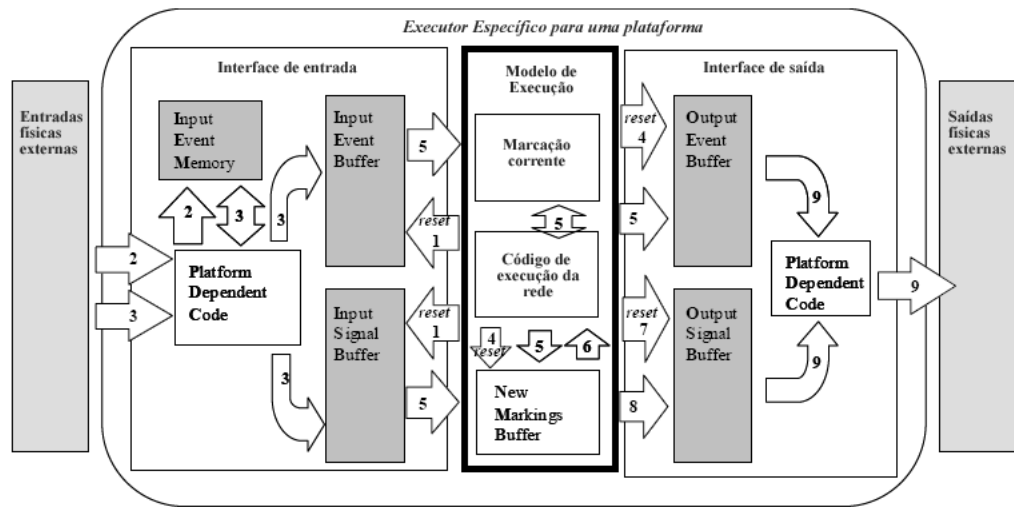


Figura 5.1- Snoopy-IOPT

#### 5.1.2- Conversor PNML2C

O conversor PNML2C faz a tradução do modelo de uma RdP descrita através do PNML para a linguagem C, podendo ser assim implementado em diversas plataformas, nomeadamente microcontroladores. Esta ferramenta, tirando partido das propriedades e características das RdP IOPT, implementa uma arquitectura de modo a que a integração num sistema real seja facilitado. Esta arquitectura é composta por

três partes fundamentais. Uma parte de entradas, outra parte em que o modelo evolui e ainda a parte das saídas. Esta arquitectura é apresentada na Figura 5.2.



**Figura 5.2** – Arquitectura do código gerado, extraído de [Pais, 04]

É ainda de referir que à data da realização desta dissertação, esta ferramenta encontra-se ainda em versões preliminares (beta), apresentando ainda alguns problemas e falhas.

Este trabalho foi iniciado em [Pais, 04], [Gomes et. al, 04] e desde então tem sido modificado no âmbito do projecto FORDESIGN, sendo uma parte integrante do mesmo como apresentado na Figura 3.31.

## 5.2- Ferramentas de desenvolvimento

### 5.2.1- MPLAB e Compilador

O MPLAB IDE é um ambiente integrado para desenvolvimento de software (Integrated Development Environment - IDE) disponibilizado pela Microchip para o desenvolvimento de aplicações para todas as famílias de microcontroladores PIC. Este possibilita e auxilia a edição de código, assim como fornece uma variedade de ferramentas de depuração (*debug*) dos programas desenvolvidos. A simulação é possível para a maioria dos dispositivos.

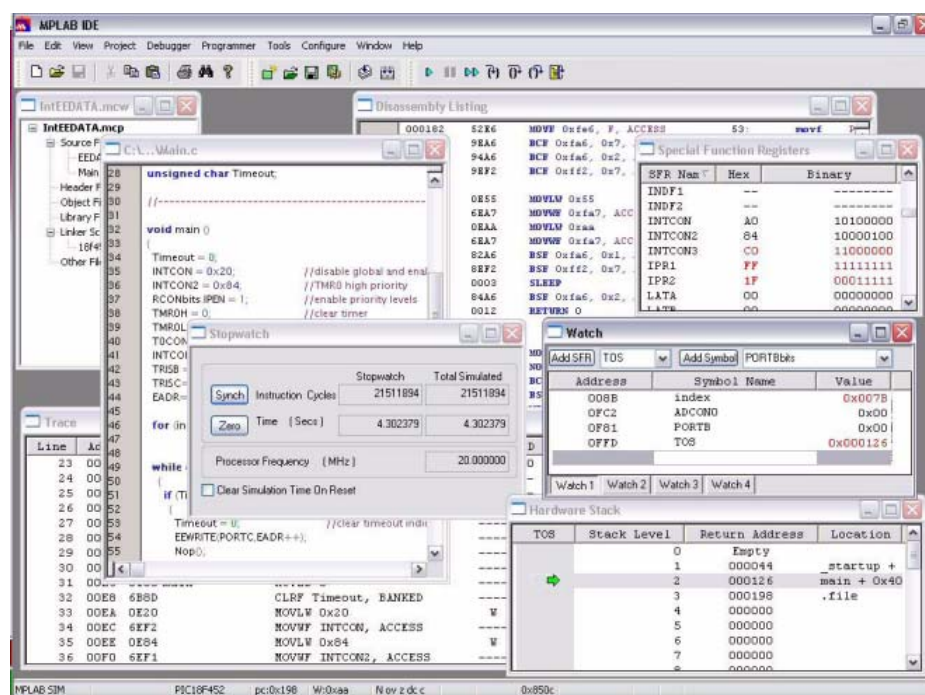


Figura 5.3 – MPLAB IDE, extraído de [Microchip, 09]

Através deste é integrada toda a programação dos dispositivos assim como o *debug* no próprio dispositivo (In-Circuit Debug - ICD). Fornece um método poderoso e extremamente útil de desenvolver os programas nos sistemas reais, com acesso e visibilidade a todas as partes do microcontrolador. Na Figura 5.3 mostra-se a sua apresentação gráfica.

Na compilação do código, este IDE invoca os compiladores que forem desejados, sejam da própria Microchip, ou de outras fabricantes como IAR, CCS, HI-TEC, entre

muitos outros. A Microchip disponibiliza para a linguagem C os compiladores C18 para a família PIC18, C30 para os PIC24 e dsPIC, e ainda o C32 para os PIC32 [Microchip, 09].

Neste trabalho a versão utilizada do MPLAB foi a 8.14, do compilador C30 foi a versão 3.01 e do C18 foi a 3.11.

### 5.2.2- Stack TCP-IP

A Microchip disponibiliza uma pilha (*stack*) de protocolos otimizados para as suas famílias de microcontroladores. Esta *stack* TCP-IP permite aos programadores de sistemas usufruir dos diversos protocolos implementados sem ter que perder tempo no seu desenvolvimento.

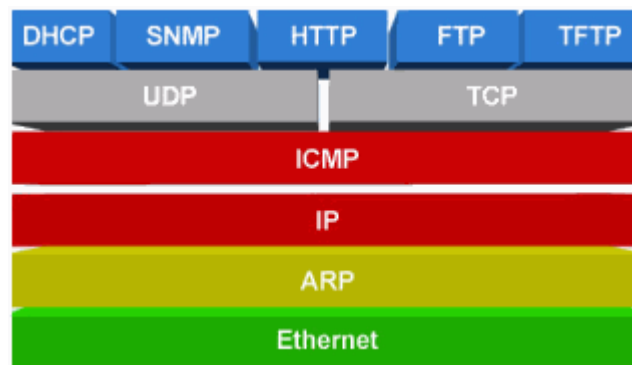


Figura 5.4 – Stack TCP-IP da Microchip, extraído de [Microchip, 09]

Na Figura 5.4 surgem os protocolos suportados por esta *stack*, mas existem mais funcionalidades que não constam, como DNS, NetBIOS Name Service, SSL, ou ainda DHCP.

No trabalho desenvolvido nesta dissertação o HTTP (Hypertext Transfer Protocol) foi a funcionalidade mais utilizada desta *stack* permitindo alojar uma página de internet com a informação do sistema.

## 5.3- Ferramentas de documentação

### 5.3.1- Doxygen

O Doxygen é uma ferramenta para geração de documentação de código. Esta pode ser usada em diversas linguagens como C++, C, Java, Objective-C, Python, Corba, Fortran, VHDL, PHP, C#, entre outras.

Esta ferramenta tem um funcionamento muito semelhante ao Javadoc, que consiste na utilização de etiquetas (*tags*) específicas para diferentes campos. Através destas *tags* inseridas em comentários no código, vai-se documentando e relacionado todo o código.

A documentação de código é algo fundamental para o mundo da programação pois facilita a legibilidade e compreensão do código feito por outros, ou mesmo por nós há algum tempo. Deste modo, facilita a passagem de código entre projectos com pessoas diferentes sendo, por vezes, algo exigido no mundo empresarial.

Esta ferramenta permite a geração da documentação em formato HTML facilitando a distribuição da documentação *online*, ou ainda no formato LaTeX. Este último é um formato muito usado na documentação científica ou profissional e permite ainda a geração da documentação para o formato PDF.

## 5.4- Ferramentas de Debug

### 5.4.1- Wireshark

O Wireshark (antigo Ethereal) é uma conhecida ferramenta de monitorização de rede (*sniffer*) que permite a captura de pacotes recebidos e enviados por um controlador de rede.

Esta ferramenta permite não só a captura, como a “dissecação” dos vários campos de cada protocolo e ainda diversas análises e ferramentas de diagnóstico. Dá suporte aos protocolos mais utilizados e conhecidos, mas também a muitos outros.

Na Figura 5.5 apresenta-se um exemplo de uma captura utilizando esta ferramenta.



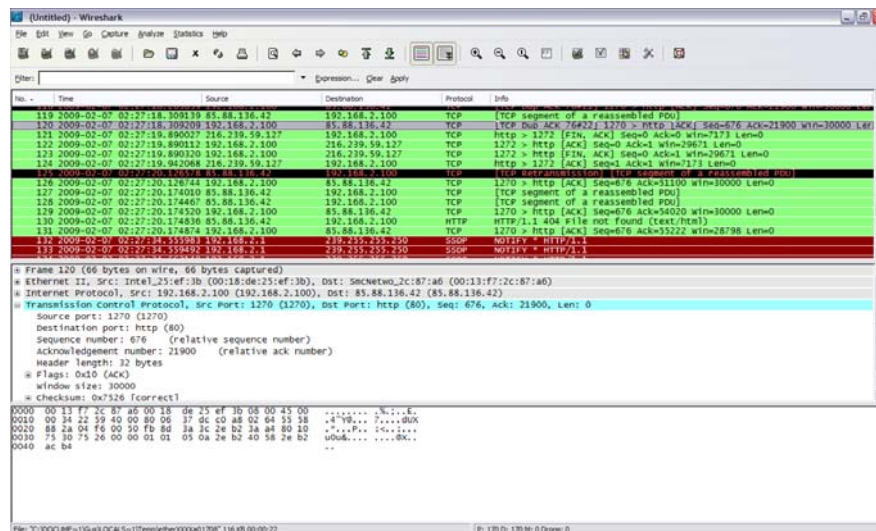


Figura 5.5 – Captura no Wireshark

### 5.4.2- ZENA Network Analyzer

O ZENA Network Analyzer da Microchip é também um *sniffer* mas para comunicações sem fios (*wireless*) que sigam a norma IEEE 802.15.4. Este analisador de rede utiliza uma placa que captura os pacotes e os envia por USB para serem mostrados numa aplicação específica.

É particularmente útil no desenvolvimento de redes sem fios tipo ZigBee ou MiWi. Esta ferramenta permite não só a “dissecação” dos pacotes presentes no ar mas fornece ainda um ambiente gráfico que permite ver as relações entre os nós.

Na Figura 5.6 apresenta-se a placa que efectua a captura, e na Figura 5.7 um exemplo das relações entre os nós sobre a planta de uma casa.



Figura 5.6 – Placa Zena Network Analyzer, extraído de [Microchip, 09]

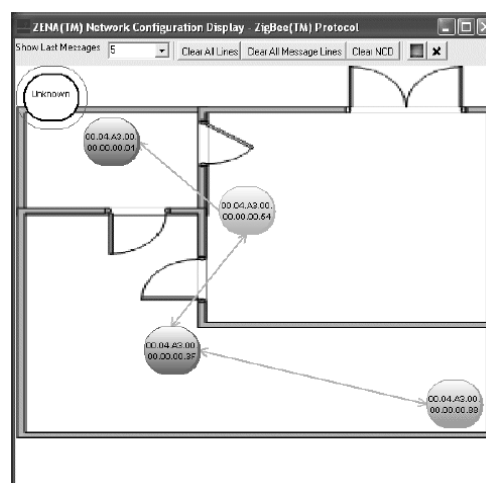


Figura 5.7 – Interface gráfico do ZENA Network Analyzer, extraído de [Microchip, 09]

## 5.5- Hardware Utilizado

### 5.5.1- PIC24FJ128GA010

O PIC24 é uma família de microcontroladores relativamente recentes, de 16 bits, com ritmos de 16 ou 40 MIPS, dispondo de inúmeros periféricos. São microcontroladores de funções gerais (*general purpose*) e contam com diversas inovações recentemente introduzidas pela Microchip como o PSV (Program Space Visibility) ou mesmo o PPS (Peripheral Pin Select). A Figura 5.8 mostra o seu diagrama de blocos

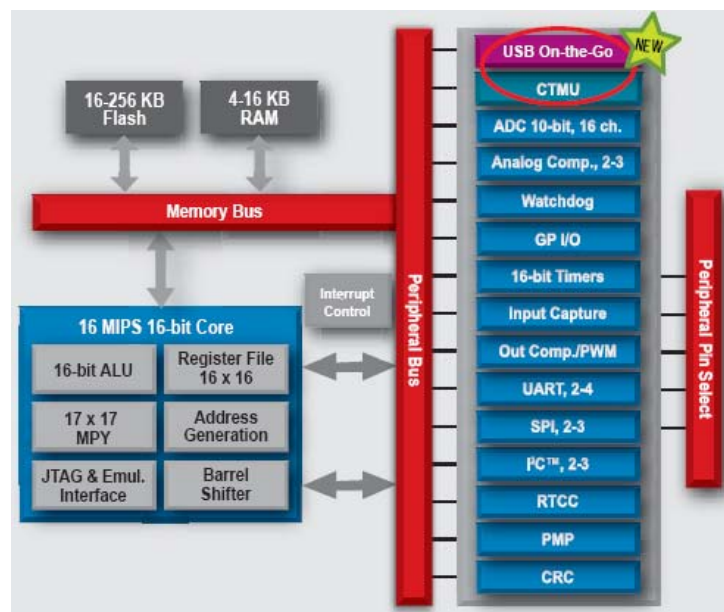


Figura 5.8 – Diagrama de blocos PIC24F, extraído de [Microchip, 09]

### 5.5.2- Placa Explorer16

A placa Explorer 16 é uma placa de desenvolvimento de baixo custo, que permite a utilização das famílias mais recentes da Microchip como os PIC 24, dsPIC e PIC32. Esta disponibiliza alguns componentes para desenvolvimento e teste como interruptores, leds, um potenciômetro, *display* LCD (*Liquid Crystal Display*), sensor de temperatura e interface série para comunicações. A sua versatilidade reside no modo como permite a fácil montagem dos diversos microcontroladores através de PIMs (Plug-In Module), ou seja, cada microcontrolador é montado numa placa/módulo que tem sempre o mesmo

esquema de pinos (*pinout*), independentemente do número de pinos do microcontrolador. Na Figura 5.9 apresenta-se um PIM do PIC32.



Figura 5.9 – PIC32 PIM, extraído de [Microchip, 09]

Outra vantagem desta placa é a existência de ranhuras (*slots*) para placas de expansão da Microchip chamadas PICTail. Estas placas permitem acrescentar diversas funcionalidades à placa como zona de prototipagem, QVGA (Quarter Video Graphics Array), controlador Ethernet, controlador RF, interface USB, IrDA, entre outras, utilizando linhas de comunicação como SPI (Serial Peripheral Interface).

A placa Explorer 16, apresentada na Figura 5.10, é assim a placa de desenvolvimento mais versátil e expansível que a Microchip disponibiliza.

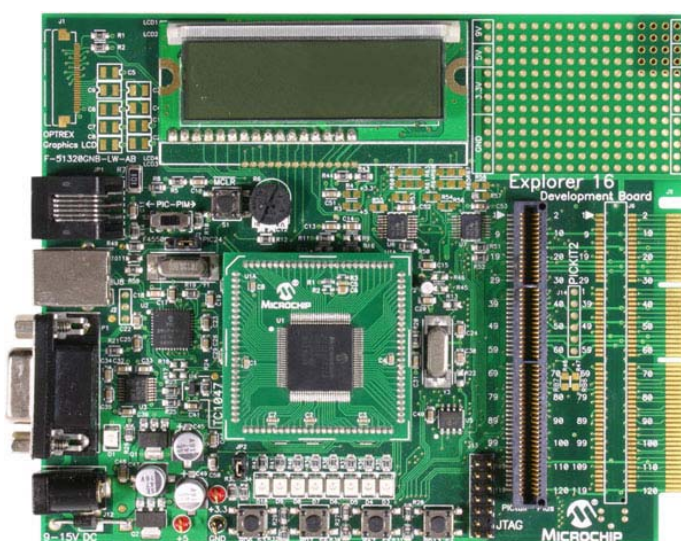


Figura 5.10 – Explorer 16, extraído de [Microchip, 09]

### 5.5.3- Kit PICDEM Z

O Kit PICDEM Z é um Kit de demonstração e desenvolvimento para aplicações sem fios sobre a norma IEEE 802.15.4. Este kit, apresentado na Figura 5.11, é composto por duas placas com dois microcontroladores PIC18F4620, dois transceptores, e ainda o

analisador de rede ZENA. Este kit permite a implementação de uma rede simples de dois nós, fornecendo assim um esquema de validação e desenvolvimento de sistemas de comunicação sem fios, contando com a ajuda do analisador de rede ZENA.



Figura 5.11 – Kit PICDEM Z, extraído de [Microchip, 09]

#### 5.5.4- Ethernet PICtail Plus

Esta placa de expansão, mostrada na Figura 5.12, permite desenvolver aplicações com comunicação por Ethernet. Vem incorporada com uma ficha para ligação do cabo rede (Ethernet) e com o controlador de rede ENC28J60, que implementa a camada MAC e PHY seguindo a norma IEEE 802.3 (Ethernet). O interface com o microcontrolador é feito através de SPI.

Utilizando esta placa em conjunto com a Stack TCP-IP é possível desenvolver diversos sistemas, tirando partido da comunicação por Ethernet e criar aplicações desenhadas para a *Web*.



Figura 5.12 – Ethernet PICtail Plus, extraído de [Microchip, 09]

### 5.5.5- PICtail Plus 2.4GHz RF

A placa de expansão RF permite adicionar ao nosso sistema a capacidade de comunicação RF, baseada na norma IEEE 802.15.4. Esta placa incorpora um módulo transceptor (*transceiver*) e uma antena como apresentado na Figura 5.13. Este módulo, o MRF24J40MA, como no PICtail anterior, implementa as camadas PHY e MAC sendo que o seu interface com o microcontrolador, é feito por SPI. Através desta placa o nosso sistema ganha a capacidade de comunicações RF, sem o esforço de desenvolvimento envolvido nesta área, que é bastante elevado. Esta placa já vem certificada sendo uma mais-valia para muitos dos fabricantes.



Figura 5.13 - PICtail Plus 2.4GHz RF, extraído de [Microchip, 09]

### 5.5.6- Comando IrDA – KBC56A

Este comando da Noritake, na Figura 5.14, permite o envio por IrDA de qualquer código da tabela ASCII (American Standard Code for Information Interchange) para o seu receptor apresentado na Figura 5.15. Por sua vez, o receptor, ligado fisicamente ao sistema, permite diversos modos de comunicação como SPI, RS232 ou RS485, cada um com configurações diferentes.

Este comando permite adicionar, de um modo muito versátil e prático, a entrada de comandos ao nosso sistema.

Torna-se extremamente útil na fase de desenvolvimento, pois podemos introduzir centenas de comandos diferentes ao nosso sistema, em vez de ter que montar diversos interruptores, para simular entradas e comportamentos. Sem ser na fase de desenvolvimento, permite criar um interface a partir do qual o utilizador do sistema interage, sendo mais cómodo e versátil do que vários interruptores, com comportamentos diferentes conforme a zona do menu [Noritake, 09].





**Figura 5.15** – Comando KBC56A, extraído de [Noritake, 09]



**Figura 5.14** - Receptor KBC56A, extraído de [Noritake, 09]

### 5.5.7- ICD2

O ICD2 é um depurador no próprio circuito (In-Circuit Debugger) que permite tanto a programação do microcontrolador como o seu *debug*. Este permite a execução do código linha a linha, assim como *breakpoints* quando se corre o programa no próprio circuito. Através destas funcionalidades temos acesso a todas as zonas do microcontrolador sejam registos, memória do programa ou variáveis (RAM). Para isto o próprio ICD2 tem que ter alguns recursos físicos de modo a pausar o programa e transferir os dados que são requeridos pelo programador. Este depurador/programador é apresentado na Figura 5.16.

Com a grande evolução recente dos microcontrolador da Microchip, os PIC24 de 16-bit, os PIC32 de 32-bits, e o crescente aumento da memória de programa, este ICD torna-se um pouco lento e daí o aparecimento do novo ICD3. Este já tem ritmos bastante mais elevados e oferece muitos mais recursos, disponibilizando, por exemplo, um maior número de *breakpoints*.



**Figura 5.16** – Microchip ICD2, extraído de [Microchip, 09]

## 6- Contribuições

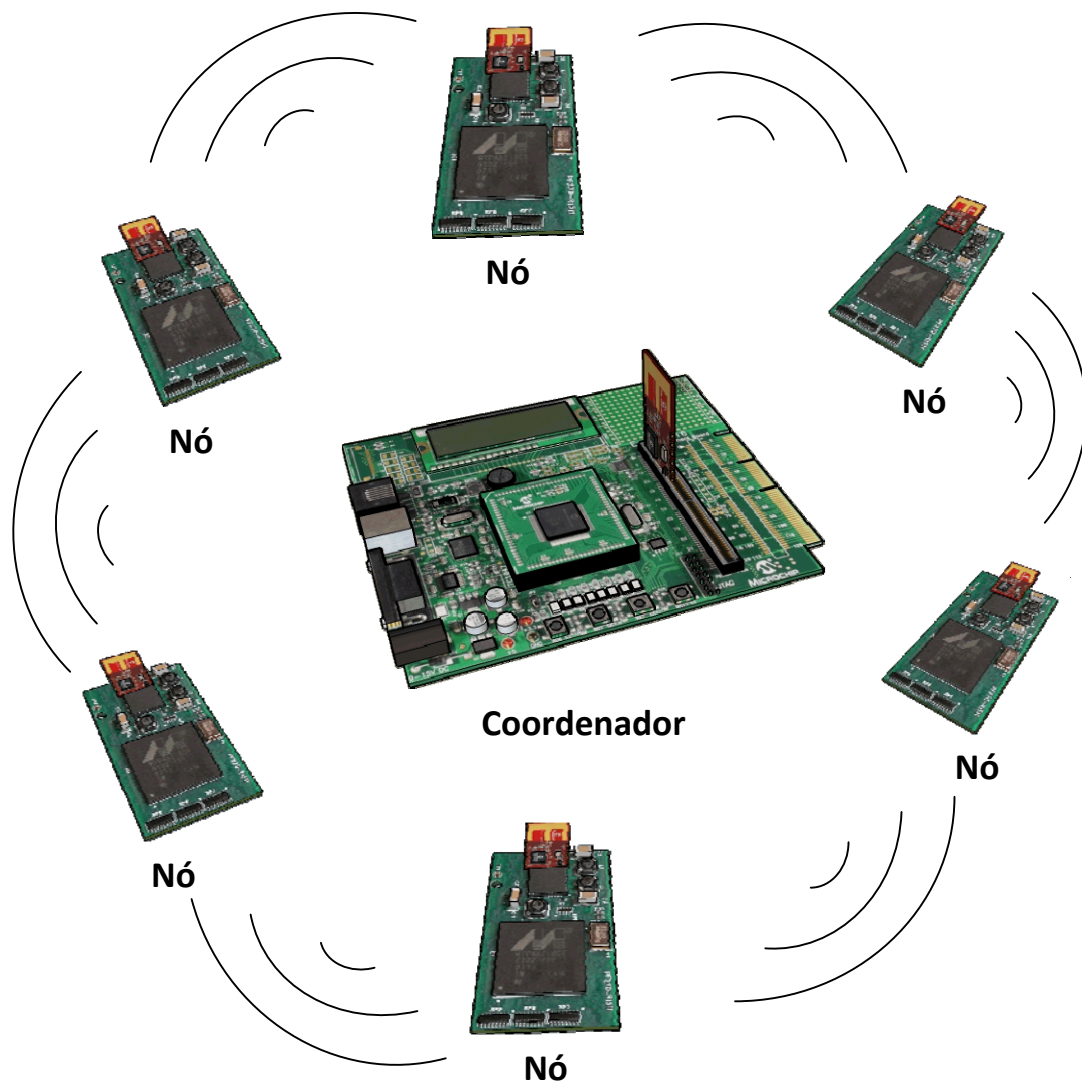
### 6.1- Descrição do Sistema

Como referido anteriormente, esta dissertação tem como objectivo o desenvolvimento de uma rede de sensores-actuadores, capazes de desempenhar funções quer no âmbito da domótica quer no âmbito das WSN. Este sistema servirá ainda como plataforma de validação do gerador de código C a partir do PNML.

Esta rede, por efeitos de simplicidade, irá ter a topologia em estrela, mencionada anteriormente na secção 1.2 desta dissertação. Deste modo todos os nós estarão ao alcance rádio do coordenador, não havendo assim a necessidade de reencaminhar mensagens. Nesta rede haverá um coordenador de rede, no qual será implementada toda a “inteligência”, controlo, decisão e gestão da rede. Por sua vez, os nós da rede serão modelados através das RdP e utilizando o PNML associado, será gerado o código C a implementar nos microcontroladores.

O sistema deverá ter a capacidade de recolher informações dos seus sensores e actuar em situações/condições específicas, deverá ser capaz de executar tarefas agendadas, e ainda disponibilizar toda a informação do estado do sistema remotamente através de uma página *Web*. Esta página *Web* será o interface com o utilizador, permitindo a configuração de todo o sistema, assim como execução de tarefas manualmente.

A topologia de rede será a apresentada na Figura 6.1. Pode-se observar o coordenador no centro da rede de modo a alcançar todos os nós. Os nós irão capturar informações dos seus sensores, enviando-as para o coordenador. Estes irão ainda accionar os seus actuadores quando o coordenador assim o ordenar.



**Figura 6.1** – Topologia da rede

O esquema de interações entre estes elementos será efectuado através de uma troca de mensagens. Estas interações serão especificadas através de um protocolo. Para se ter uma ideia destas interações, a Figura 6.2 demonstra o tipo de comunicação entre estes elementos. O protocolo em si será abordado mais à frente. É de referir ainda que nesta rede o coordenador toma o papel principal, sendo este quase sempre a requerer uma resposta dos nós. Só no início, aquando do registo dos nós, é que estes enviam uma mensagem sem que lhes tenha sido solicitada.



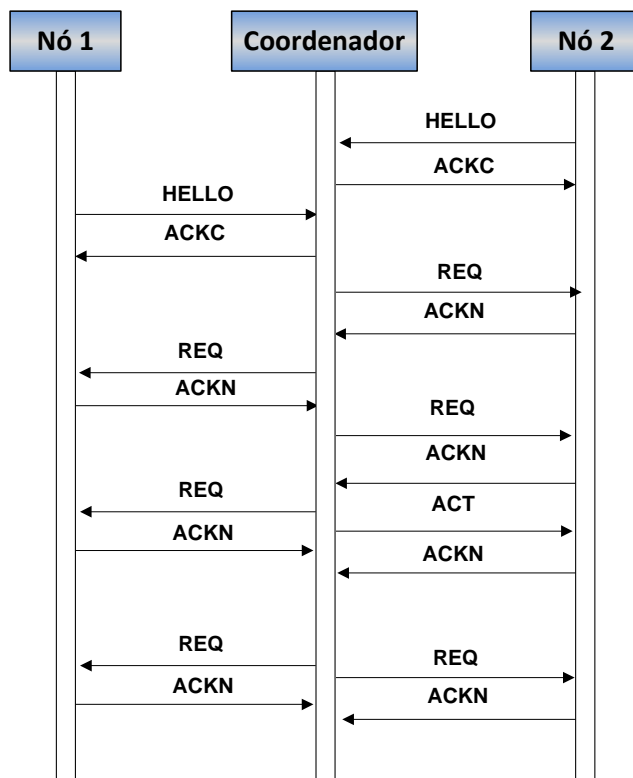


Figura 6.2 – Diagrama de sequência do Protocolo

## 6.2- Casos de uso

De modo a fornecer uma melhor compreensão e especificação do sistema, apresenta-se o diagrama de casos de uso.

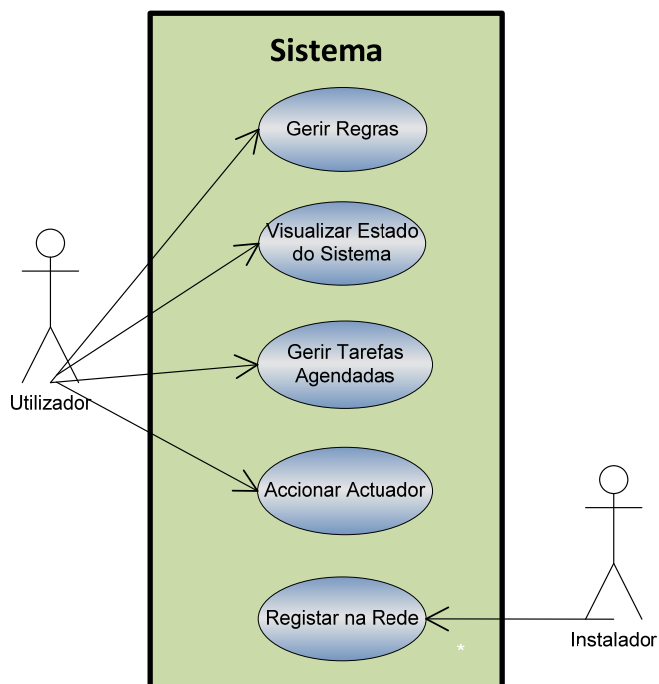


Figura 6.3 – Casos de uso do Sistema

Do ponto de vista do sistema na sua globalidade, as interacções existentes com o mundo externo são as apresentadas no diagrama de casos de uso da Figura 6.3. Assim, vendo o coordenador e os diversos nós como um só sistema, os casos de uso são a gestão de regras (adicionar, remover e editar regras), a gestão de tarefas agendadas, a obtenção de informação sobre o estado do sistema e o accionamento de um actuador existente na rede. Nos casos de uso referidos o actor é o utilizador da rede. Cada nó tem ainda o papel de actor (operado pelo instalador) no momento em que se registam, pois é por iniciativa dos nós que uma série de acontecimentos é desencadeada.

Estes conceitos de regras e tarefas agendadas irão ser explicados mais à frente.

Por outro lado, os dois subsistemas podem ser vistos isoladamente, de modo a identificar os casos de uso especificamente do coordenador e dos nós.

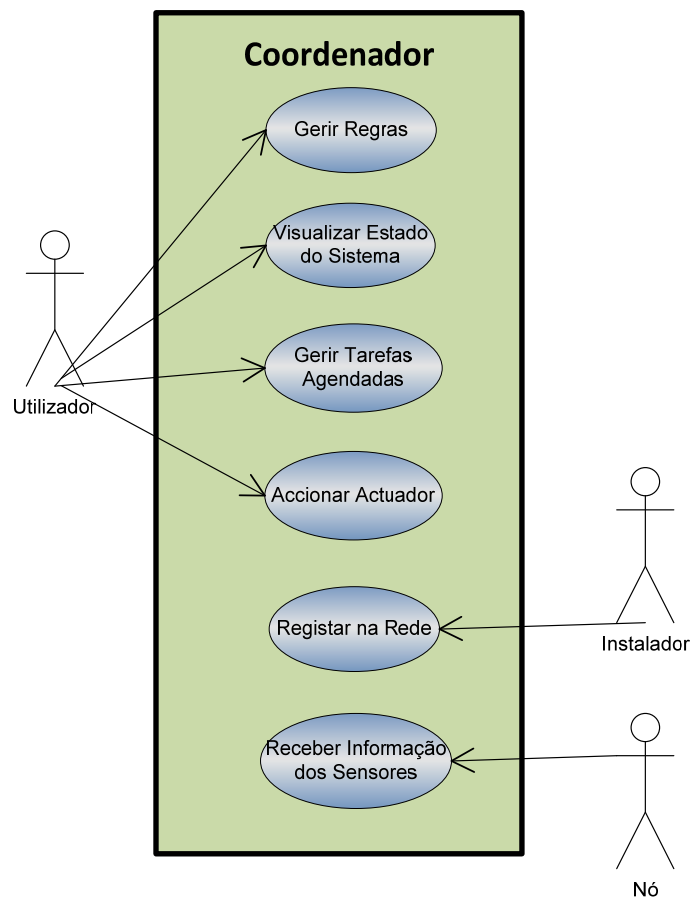


Figura 6.4 – Casos de uso do coordenador

Os casos de uso do coordenador, apresentados na Figura 6.4, são muito semelhantes aos do sistema, pois é neste que reside toda a inteligência e praticamente toda a interacção com o utilizador. Neste diagrama de casos de uso surge ainda a recepção de informação dos sensores proveniente dos nós, que na visão do sistema global não era tido em conta.

Por sua vez, os nós têm o diagrama de casos de uso apresentado na Figura 6.5. Neste caso a maioria de interacções é por parte do coordenador, uma vez que o instalador apenas indica quando o nó se deve registar.

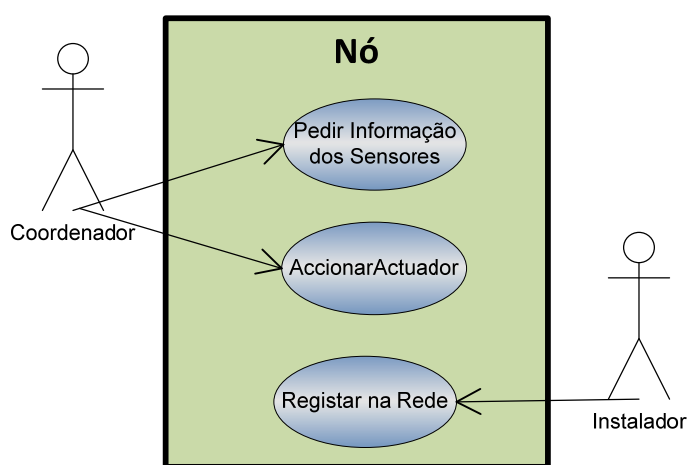


Figura 6.5 – Casos de uso dos nós

### 6.3- Arquitectura do Sistema

Antes de se abordar a arquitectura implementada neste trabalho, baseado no diagrama de casos de uso e nos objectivos, identifica-se os requisitos do sistema e como os colmatar.

Como referido anteriormente, o sistema deverá ser capaz de agendar tarefas, ou seja, o utilizador poderá accionar um actuador qualquer a uma hora e data específica.

Para este efeito será necessário implementar um módulo que guarde essas tarefas e, que monitorizando a referência temporal do sistema, faça despoletar essas tarefas na altura devida.

Outro dos requisitos é a capacidade do sistema reagir automaticamente. O sistema deverá, ao monitorizar os sensores, ser capaz de accionar mecanismos em situações

específicas, sem que para isso o utilizador tenha que intervir. Deste modo o utilizador deverá criar um conjunto de regras definindo que em situações específicas (por exemplo, o sensor de temperatura acima de 30°), algum actuador seja accionado (por exemplo, ligar o ar condicionado - AC). Assim, deverá haver um módulo que aloje estas regras, e que esteja constantemente a monitorizar os valores dos sensores, de modo a que possa reagir nas situações previstas.

Em seguida apresentam-se os modelos e as arquitecturas desenvolvidas tanto para o coordenador como para os nós.

### **6.3.1- Coordenador**

Em termos de *hardware* a plataforma utilizada no desenvolvimento do sistema foi a placa Explorer 16 com o PIM do PIC24FJ128GA010 e duas placas de expansão PICtail, uma de Ethernet e outra de RF. Foi ainda necessário adicionar mais uma *slot* de expansão à Explorer 16, pois esta apenas vem com uma de fábrica. Embora haja compatibilidade dos pinos para as duas PICtail, fisicamente é impossível montar as duas na mesma *slot*, isto porque os pinos atribuídos ao SPI estão seguidos na *slot* e a placa de circuito impressa (*Printed Circuit Board* - PCB) de cada PICtail sobrepõe os pinos da outra. Na Figura 6.6 apresenta-se a placa ligada ao programador/depurador ICD2 e ainda com o cabo cruzado (*crossover*) ligado ao PICtail Ethernet.

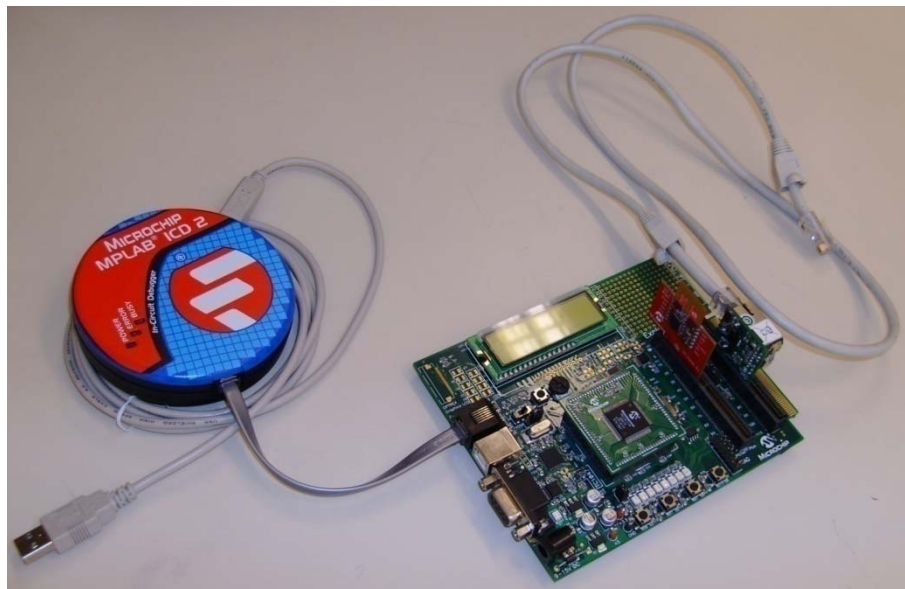


Figura 6.6 – Hardware do Coordenador

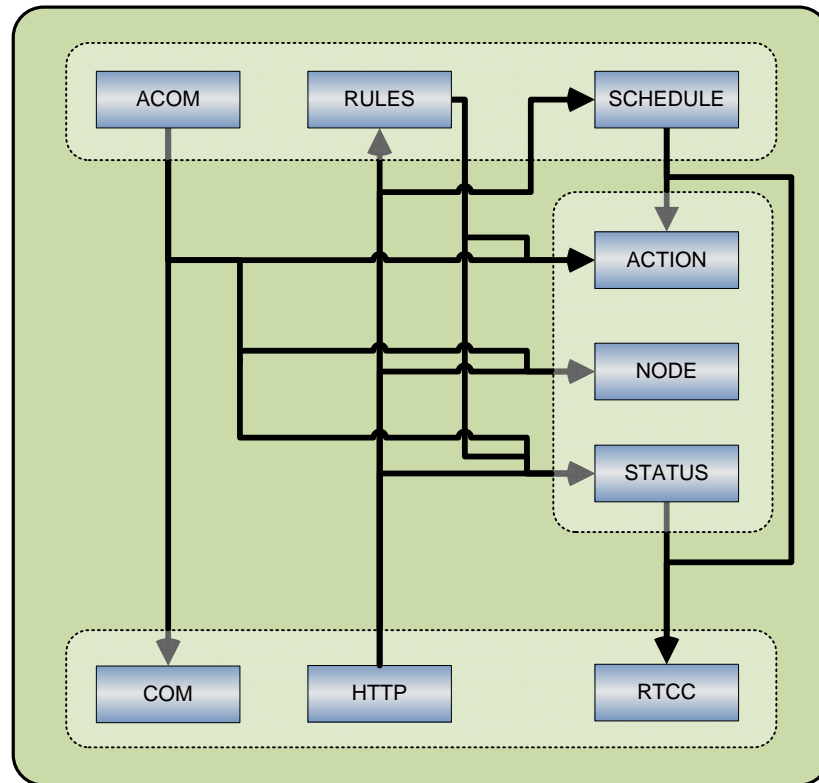
As questões de modelação e arquitectura desenvolvida para a concretização do coordenador da rede serão abordadas em seguida. Na Figura 6.7 apresenta-se o diagrama de blocos/módulos do coordenador de rede.

O UML não foi utilizado para esta representação, pois grande parte das suas representações são direccionadas ao objecto. Na implementação deste sistema foi utilizada a linguagem C, que não é orientada ao objecto, sendo mais virada para os processos. Embora na maioria dos módulos implementados sejam definidos tipos específicos de variáveis (como numa classe), utilizando *typedef*, estes não são na verdade objectos como nas linguagens orientadas ao objecto. Deste modo, a arquitectura foi expressa através de um diagrama de blocos/módulos, que mostra as relações entre estes.

Pode-se observar na Figura 6.7 três agrupamentos/camadas diferentes. O agrupamento mais em baixo reúne os módulos de baixo nível que interagem directamente com o *hardware*. O conjunto do meio é onde estão alojadas as informações do sistema que irão fazer o sistema evoluir, formando uma espécie de base de dados para o sistema. Todo o processamento, decisão e evolução do sistema é despoletado na camada mais em cima, sendo aqui que todas as decisões são realizadas, baseando-se na camada intermédia e agindo na camada mais baixa.

Deste modo é possível identificar quais os módulos que conferem todo o controlo e decisão ao sistema permitindo a automação do mesmo. É nos módulos Schedule e

Rules que os mecanismos autónomos são implementados. É apenas nestes dois que são geradas as acções da rede, quer por avaliação de situações predefinidas quer por tarefas agendadas. Já no módulo Acom está implementado o protocolo e é neste que as evoluções e interacções de rede estão descritas.



**Figura 6.7** – Diagrama de blocos do coordenador

Na implementação do sistema, considerou-se a utilização de sistemas operativos que fornecessem um ambiente estruturado ao desenvolvimento como o TinyOS [TinyOS, 09], ou mesmo o TinyGALS [Cheong et al, 03]. Essas plataformas não foram utilizadas por alguns motivos. Houve desde o início deste trabalho a vontade de modelar e implementar este complexo sistema de raiz, permitindo o acesso e o controlo total do sistema. Outro motivo é o facto de estas plataformas introduzirem um acréscimo no tamanho do código, recurso que é limitado. O tamanho do código foi um factor crucial no desenvolvimento, obrigando a mudança de microcontrolador a meio do projecto, migrando do PIC24FJ64GA004 com 64 KBytes para o PIC24FJ128GA010 com 128 KBytes. Esta necessidade surgiu quando a *Stack* TCP-IP foi adicionada ao projecto, que

mesmo com apenas as funcionalidades mínimas e necessárias, ocupa uma boa parte da memória de programa.

Embora não utilizada a plataforma do TinyGALS, esta serviu um pouco de exemplo à arquitectura do sistema desenvolvido, na medida em que foi implementada uma arquitectura por módulos de software, cada um com as suas funcionalidades a correr e em certas alturas trocam informação entre módulos. Claro que na arquitectura desenvolvida todas estruturas e mecanismos não foram formalmente definidos, como nas plataformas mencionadas.

Em termos dos módulos de software, apresenta-se em seguida uma descrição das suas funções. É importante referir ainda que existem outros módulos que tiveram que ser implementados, mas como são de baixo nível (controlo de *hardware*, camada intermédia, ou apenas por efeitos de teste) não são mencionados na arquitectura. Estes são por exemplo o controlo temporal por *timers*, o controlo do LCD, o controlo das comunicações série por RS232, o controlo SPI dos periféricos externos, a gestão do controlador Ethernet ou ainda do transceptor para a RF.

## RTCC

O módulo RTCC faz uso do relógio de tempo real e calendário (Real Time Clock Calendar - RTCC) existente no PIC24F.

Este RTCC, que alimentado por um cristal a 32.768 KHz, fornece uma referência temporal precisa a todo o sistema. O RTCC disponível no PIC24F disponibiliza uma referência a nível de horas mas também dia da semana, dia do mês, mês e ano, como apresentado na Tabela 6.1.



Tabela 6.1 – Formato do tempo no RTCC, adaptado de [PIC24F, 07]

Este RTCC permite ainda activar alarmes que ocorrerão no momento exacto em que for configurado, gerando uma interrupção (*interrupt*) no microcontrolador. Este mecanismo de alarme é extremamente versátil, pois permite, através de máscaras (*masks*), configurar alarmes desde todos os meios segundos a todos os anos. Este mecanismo é apresentado na Tabela 6.2.

Assim, o módulo RTCC implementado tem a função de configurar e controlar o periférico, fornecendo a todo o momento a referência temporal ao sistema. Este módulo gere ainda a configuração e utilização dos alarmes, pois só é possível activar um de cada vez, e quando este dispara atende a sua interrupção. É importante referir ainda que para tempos mais reduzidos que o segundo, foi implementado um outro módulo, que faz uso de um temporizador (*timer*) do microcontrolador, de modo a controlar o tempo na ordem dos milissegundos (ms). Esta funcionalidade foi importante, por exemplo, na implementação do protocolo, para o caso dos reenvios. Para unidades de tempo na ordem dos segundos este RTCC já apresenta resultados bastante aceitáveis.

	Dia da Semana	Mês	Dia	Horas	Minutos	Segundos
A cada meio Segundo	<input type="checkbox"/>	<input type="checkbox"/> <input type="checkbox"/> / <input type="checkbox"/> <input type="checkbox"/>	<input type="checkbox"/> <input type="checkbox"/>	<input type="checkbox"/> <input type="checkbox"/> : <input type="checkbox"/> <input type="checkbox"/> : <input type="checkbox"/> <input type="checkbox"/>		
A cada Segundo	<input type="checkbox"/>	<input type="checkbox"/> <input type="checkbox"/> / <input type="checkbox"/> <input type="checkbox"/>	<input type="checkbox"/> <input type="checkbox"/>	<input type="checkbox"/> <input type="checkbox"/> : <input type="checkbox"/> <input type="checkbox"/> : <input type="checkbox"/> <input type="checkbox"/>		
A cada 10 Segundos	<input type="checkbox"/>	<input type="checkbox"/> <input type="checkbox"/> / <input type="checkbox"/> <input type="checkbox"/>	<input type="checkbox"/> <input type="checkbox"/>	<input type="checkbox"/> <input type="checkbox"/> : <input type="checkbox"/> <input type="checkbox"/> : <input type="checkbox"/> <input type="checkbox"/>		<input type="checkbox"/> s
A cada Minuto	<input type="checkbox"/>	<input type="checkbox"/> <input type="checkbox"/> / <input type="checkbox"/> <input type="checkbox"/>	<input type="checkbox"/> <input type="checkbox"/>	<input type="checkbox"/> <input type="checkbox"/> : <input type="checkbox"/> <input type="checkbox"/> : <input type="checkbox"/> <input type="checkbox"/>		<input type="checkbox"/> s <input type="checkbox"/> s
A cada 10 Minutos	<input type="checkbox"/>	<input type="checkbox"/> <input type="checkbox"/> / <input type="checkbox"/> <input type="checkbox"/>	<input type="checkbox"/> <input type="checkbox"/>	<input type="checkbox"/> <input type="checkbox"/> : <input type="checkbox"/> m : <input type="checkbox"/> s <input type="checkbox"/> s		
A cada Hora	<input type="checkbox"/>	<input type="checkbox"/> <input type="checkbox"/> / <input type="checkbox"/> <input type="checkbox"/>	<input type="checkbox"/> <input type="checkbox"/>	<input type="checkbox"/> <input type="checkbox"/> : <input type="checkbox"/> m <input type="checkbox"/> m : <input type="checkbox"/> s <input type="checkbox"/> s		
A cada Dia	<input type="checkbox"/>	<input type="checkbox"/> <input type="checkbox"/> / <input type="checkbox"/> <input type="checkbox"/>	<input type="checkbox"/> <input type="checkbox"/>	<input type="checkbox"/> h <input type="checkbox"/> h : <input type="checkbox"/> m <input type="checkbox"/> m : <input type="checkbox"/> s <input type="checkbox"/> s		
A cada Semana	<input type="checkbox"/> d	<input type="checkbox"/> <input type="checkbox"/> / <input type="checkbox"/> <input type="checkbox"/>	<input type="checkbox"/> <input type="checkbox"/>	<input type="checkbox"/> h <input type="checkbox"/> h : <input type="checkbox"/> m <input type="checkbox"/> m : <input type="checkbox"/> s <input type="checkbox"/> s		
A cada Mês	<input type="checkbox"/>	<input type="checkbox"/> <input type="checkbox"/> / <input type="checkbox"/> d <input type="checkbox"/> d	<input type="checkbox"/> <input type="checkbox"/>	<input type="checkbox"/> h <input type="checkbox"/> h : <input type="checkbox"/> m <input type="checkbox"/> m : <input type="checkbox"/> s <input type="checkbox"/> s		
A cada Ano	<input type="checkbox"/>	<input type="checkbox"/> m <input type="checkbox"/> m / <input type="checkbox"/> d <input type="checkbox"/> d	<input type="checkbox"/> <input type="checkbox"/>	<input type="checkbox"/> h <input type="checkbox"/> h : <input type="checkbox"/> m <input type="checkbox"/> m : <input type="checkbox"/> s <input type="checkbox"/> s		

**Tabela 6.2** – Máscaras de Alarme do RTCC, adaptado de [PIC24F, 07]



## Status

O módulo Status é onde estão alojadas todas as informações dos sensores espalhados na rede. Neste reside uma lista que contém a informação actualizada de cada sensor, que é consultada pelos outros módulos de modo a que o sistema possa reagir a situações predefinidas (implementado no módulo Regras).

São adicionados itens à lista aquando do registo de um novo nó na rede. De modo a saber a quantidade e o tipo de sensores desse nó a adicionar à lista, é verificado o seu tipo. É necessário ainda que a informação associada a cada tipo de nó exista previamente no coordenador. Este processo permite o desenvolvimento de novos tipos de nós, sendo apenas necessário a actualização do *firmware* do coordenador e é completamente transparente para o resto dos elementos da rede.

Cada item desta lista tem associada uma referência temporal (*timestamp*) que marca a ultima actualização desse sensor. Este módulo está constantemente a percorrer a lista, verificando se existem sensores que não são actualizados há mais que um tempo predefinido, comparando o *timestamp* com o tempo actual. Ao detectar itens desactualizados marca-os, de modo a que o sistema possa depois detectar que é necessário interrogar esse nó por informações.

## Nodes

Tanto o módulo Nodes como o Action são apenas repositórios de informação, não havendo nenhum processo a efectuar. O módulo Nodes guarda a lista dos nós presentes na rede e outros dados associados a estes, como o seu ID, tipo de nó e nome. É este módulo que, ao entrar um novo nó, lhe atribui um ID procurando o próximo ID disponível.

## Action

Como referido anteriormente, este módulo apenas aloja informação. Mais especificamente este aloja uma lista de acções. À medida que o sistema evolui ele tem a necessidade de reagir/actuar. Quando esta necessidade ocorre o sistema adiciona um item a esta lista, representando a necessidade de actuar. Como essa necessidade

de actuar pode despoletar várias acções em simultâneo, esta lista guarda-as até que o sistema as possa desencadear. Assim a necessidade de actuar ao surgir, é representada nesta lista temporariamente até que o sistema possa reagir enviando a acção ao nó. Este mecanismo é necessário pois o protocolo pode estar à espera de uma confirmação (*acknowledge*) ou noutra situação que não possa sobrepor o canal de comunicação.

É claro que este processo introduz um ligeiro atraso na reacção do sistema, mas para as áreas de aplicação do sistema este não é significativo. Ainda assim, de modo a minimizar este atraso cada acção tem uma prioridade associada. Existem quatro níveis de prioridade desde a baixa à emergência, possibilitando uma reacção do sistema conforme a necessidade real. Deste modo, mesmo ocorrendo no mesmo instante, é possível sobrepor acções mais importantes às outras. Estas prioridades são definidas pelo utilizador do sistema quando o configura.

### **Schedule**

O módulo Schedule contém uma lista que guarda as tarefas agendadas. Quando o utilizador agenda uma tarefa é nesta lista que será guardada. À medida que esta lista é preenchida este módulo é responsável por seleccionar a tarefa mais próxima, configurando o alarme do RTCC. Como RTCC só permite um alarme de cada vez, é necessária a gestão deste recurso avaliando sempre qual a tarefa da lista que ocorrerá antes de todas as outras. Assim que o alarme dispare, é gerada uma interrupção, e este módulo encarrega-se de adicionar a tarefa respectiva à lista de acções no módulo Action, deixando assim o sistema encarregar-se de a enviar. Logo a seguir, estando o alarme do RTCC livre, selecciona qual a próxima tarefa e configura o alarme no RTCC.

Este módulo poderia ter sido implementado de modo a não fazer uso do alarme do RTCC, não sendo necessária esta gestão do recurso. Em vez de se utilizar o mecanismo de interrupção, podia-se ter utilizado o mecanismo de *pooling*, estando constantemente a ver se o tempo actual é igual ou superior ao de alguma tarefa agendada. Este método não foi utilizado pois havendo um recurso do *hardware* que fornece essa função, seria pouco expedito não o utilizar.

## Rules

É neste módulo Rules que estão implementados os automatismos de reacção a eventuais situações predefinidas. Mais uma vez, existe uma lista que neste caso aloja um conjunto de regras. A função deste módulo é estar constantemente a monitorizar o estado dos sensores e avaliar se existe alguma regra que contemple essa situação, despoletando a reacção associada, adicionando-a à lista de acções.

De modo a tornar estes automatismos o mais variados possível e fornecer ao utilizador uma grande versatilidade na definição de regras foram criados três tipos de regras.

**Simples** – Estas são regras que definem directamente um valor limite de um sensor específico num nó. A partir deste valor o processo de avaliação verifica se a regra deve disparar ou não.

Um exemplo de aplicabilidade é a definição de uma regra que associa o valor de uma temperatura numa divisão específica à actuação do AC. Por exemplo pode-se definir que se a temperatura na sala estiver abaixo dos 17° o sistema deve ligar o AC nos 20°. Outra regra pode ser definida para o caso de se a temperatura estar acima dos 27°, activar de novo o AC a 20°. Deste modo pode-se estabelecer um patamar/intervalo de temperaturas para a divisão.

**Grupo** - Estas são regras que englobam todos os sensores do mesmo tipo. Com estas pode-se estabelecer uma acção, definindo apenas uma regra, que acontecerá no caso de qualquer um dos sensores verificar a condição. O processo de verificação é idêntico ao das regras simples, mas em vez de ser específico somente a um nó, estas verificam todos os nós que tenham sensores desse tipo. Assim utilizando este tipo de regra pode-se, por exemplo, definir que no caso de algum dos vários sensores de fumo disparar, activa-se um alarme.

**Condicional** – Este tipo tem um método de avaliação diferente do das anteriores. Neste não são definidos valores limites para o disparo.

Estas regras associam regras do tipo simples ou de grupo, de modo a formar uma regra condicionada às outras. Isto é, permitem criar regras baseadas no caso de alguma regra (simples ou de grupo) estar activa ou não. Deste modo é possível disparar uma regra apenas se um conjunto de situações se verificar. Por exemplo pode-se criar uma regra que avalie se a luminosidade e a temperatura numa divisão for maior que certos valores em simultâneo, esta dispara fechando os estores dessa divisão. O processo de avaliação destas regras verifica apenas duas das outras regras, mas é possível criar uma regra condicional que dispare quando uma outra condicional estiver activa e uma terceira regra simples também. Deste modo é possível associar várias regras para uma única acção.

Mesmo com esta versatilidade na definição de regras, foi ainda adicionada uma funcionalidade de modo a conferir mais possibilidades ao sistema. Definiu-se que cada regra pode, mas não é obrigatório, ter uma “proacção”. Esta é uma acção que é desencadeada a seguir à acção inicial. Funciona da seguinte maneira. No caso de uma regra disparar, uma acção é executada. Supondo que essa acção irá em oposição à grandeza que o sensor mediu e activou a regra, é de esperar que a regra passe do seu estado activo para inactivo. Então quando uma regra, que tenha associada uma “proacção” é activada e depois inactiva, esta desencadeia uma nova acção, a “proacção”. Por exemplo definindo uma regra com “proacção” é possível fazer com que quando um sensor de temperatura estiver com o valor abaixo dos 15°, uma acção irá activar um aquecedor para aumentar a temperatura. Quando a temperatura voltar a passar dos 15° uma outra acção irá desligar o aquecedor (neste exemplo o aquecedor não tem termóstato).

Na avaliação do disparo das regras existem algumas comparações possíveis. No caso de ser um valor pode-se comparar se está acima, abaixo ou igual ao valor limite. No caso de ser uma variável booleana pode-se verificar se está activo ou não. Para as regras condicionais é possível comparar o estado de duas regras aplicando algumas operações da álgebra de Boole como AND, OR, NOR e NAND.

## COM

Este módulo pretende ser uma abstracção de todas as camadas inferiores à camada de aplicação implementada no módulo ACOM. Estas camadas são baseadas no modelo OSI simplificado. Elas são a camada mais baixa (LCOM) que por SPI interage com o controlador Ethernet e RF e a mais alta (NCOM) é camada mais acima que prepara os dados para serem enviados no LCOM e implementa uma máquina de estados para o envio das mensagens aguardando pelos respectivos *acknowledges*, e reportando o sucesso ou falha do envio à camada acima, a ACOM.

A camada NCOM quando envia uma mensagem fica a aguardar o respectivo *acknowledge*. Se ao fim de um certo tempo não o receber, faz o reenvio e aguarda novamente. Este processo é repetido um certo número de vezes, até que por fim, se o *acknowledge* não chegar, reporta à camada ACOM a falha.

Do ponto de vista da camada ACOM, esta apenas sabe se o envio teve sucesso ou não, e considera que houve apenas um envio, embora a mensagem tenha sido reenviada um número predefinido de vezes.

Na concretização do sistema foram elaborados dois LCOM diferentes, um por Ethernet (utilizando o User Datagram Protocol - UDP) e outro por RF. A solução actual do sistema trabalha por RF, mas se for desejado migrar a solução para uma rede agora ligada por Ethernet será apenas necessário uma pequena alteração na camada NCOM que em vez de chamar a LCOM por RF passaria a usar a LCOM por Ethernet. Deste modo todo o sistema desenvolvido pode utilizar dois meios de comunicação tornando-o muito versátil e do ponto de vista comercial, se fosse um produto, poderia ser uma grande vantagem. A programação por camadas/módulos permite uma abstracção total do que existe nas outras, sendo apenas necessário definir as interacções entre as camadas, trazendo grandes benefícios tanto ao sistema em si, mas também a quem o desenvolve e mantém ou actualiza.

A comunicação através de Ethernet foi muito utilizada na fase de desenvolvimento para testes e validação do correcto funcionamento do protocolo e do sistema na sua totalidade. O protocolo utilizado foi o UDP por ser semelhante ao que acontece na RF, uma vez as tramas RF são mensagens (*datagrams*) transmitidas sem garantia de

recepção. Para os testes por Ethernet, como abordado mais à frente, foi criada uma aplicação em JAVA que simula o comportamento dos nós e permitiu testar todo o funcionamento do coordenador que comporta esta arquitectura complexa.

## **ACOM**

O módulo ACOM é a camada aplicação onde o protocolo de comunicações foi implementado. De modo a seja perceptível o seu funcionamento este será explicado mais à frente quando se abordar o protocolo desenvolvido.

## **HTTP**

O módulo HTTP como o COM representa uma abstracção de alguns módulos de software que implementam diversas funcionalidades. O HTTP engloba funcionalidades como o TCP, UDP, ARP, ICMP, IP, DHCP, HTTP entre outras. Estas funcionalidades foram conferidas ao sistema através da *Stack* TCP-IP que a Microchip fornece.

Deu-se o nome de HTTP ao módulo pois é graças a este que é possível alojar uma página *Web* dentro deste sistema, de modo a fornecer um interface com utilizador extremamente apelativo e de forma remota. Foi para adicionar estas funcionalidades ao sistema que toda a *Stack* TCP-IP foi utilizada neste trabalho.

A concretização do interface com o utilizador através da página *Web* foi também parte significativa do desenvolvimento do trabalho, pois foi utilizado HTML para a página, AJAX (Asynchronous Javascript And XML) e JavaScript para mecanismos de processamento e dinamismo da página, XML para alojar informações do sistema, SVG (Scalable Vectorial Graphics) na apresentação de gráficos dinâmicos na página e ainda CSS (Cascading Style Sheets) para o grafismo da página.

A Microchip criou um sistema de ficheiros chamado MPFS (Microchip File System) e é neste formato que os dados são guardados na EEPROM. Para programar a EEPROM com estes ficheiros utiliza-se a ferramenta MPFS que vem com a *Stack*.

A grande versatilidade desta *Stack* reside no mecanismo de substituição. Através deste mecanismo é possível adicionar dinamismo ao HTML e a outros formatos estáticos. Por exemplo num ficheiro HTML estático, coloca-se um marcador específico

(qualquer palavra rodeada de ~, por exemplo ~hello~). Esta página ao ser acedida através da *Stack* irá executar a chamada duma função que reside no microcontrolador (criada automaticamente, com o nome HTTPPrint\_hello), sendo possível escrever a qualquer mensagem no lugar do marcador. Do lado do navegador de internet (*browser*) este nunca lê o marcador ~...~, apenas o valor que o microcontrolador imprime na altura. É assim que, usando esta *Stack*, é possível criar páginas com conteúdo dinâmico, ao contrário do usual método de aceder a uma base de dados.

Neste trabalho este mecanismo foi intensamente utilizado, até mesmo imprimindo código JavaScript que o *browser* sabe interpretar de modo a preencher *arrays*, entre outras técnicas. Já para a geração dos gráficos dinâmicos por SVG, foi também utilizada esta técnica, fazendo com que os gráficos estejam actualizados.

O JavaScript foi muito utilizado porque é código que o *browser* sabe interpretar, aliviando assim o trabalho do microcontrolador, pois este código é executado pelo processador do visitante, e não no servidor da página, que neste caso é um microcontrolador limitado. A utilização do JavaScript permitiu criar páginas que são modificadas a cada evento, permitiu a criação de listas e tabelas com tamanhos dinâmicos, entre outras funcionalidades.

A tecnologia AJAX é utilizada para efectuar alterações na página, sem que para isso seja necessário a actualizar. Depois de tabelas serem impressas pelo JavaScript, cujos *arrays* foram preenchidos pelo método de substituição da *Stack*, o AJAX vai alterar os valores numéricos como tipos de nós ou dispositivos por textos que residem num XML. Deste modo a informação que é apresentada na página está centralizada no XML, sendo por isso extremamente fácil a sua modificação, sem ser necessário a edição de todos os ficheiros HTML.

Já o CSS foi utilizado para criar uma apresentação apelativa da página e de forma a ser dinâmica na medida em que todos parâmetros gráficos estão guardados neste ficheiro.

Foi ainda um esforço constante conseguir criar este interface apelativo, dinâmico e robusto, com apenas 256 kbit disponíveis na EEPROM. Todas estas funcionalidades, de certo modo avançadas, foram concretizadas sobre um microcontrolador limitado, o PIC24F.

Depois de explicados todos os módulos do coordenador em separado, expressa-se o funcionamento genérico do coordenador de rede através do fluxograma da Figura 6.8.

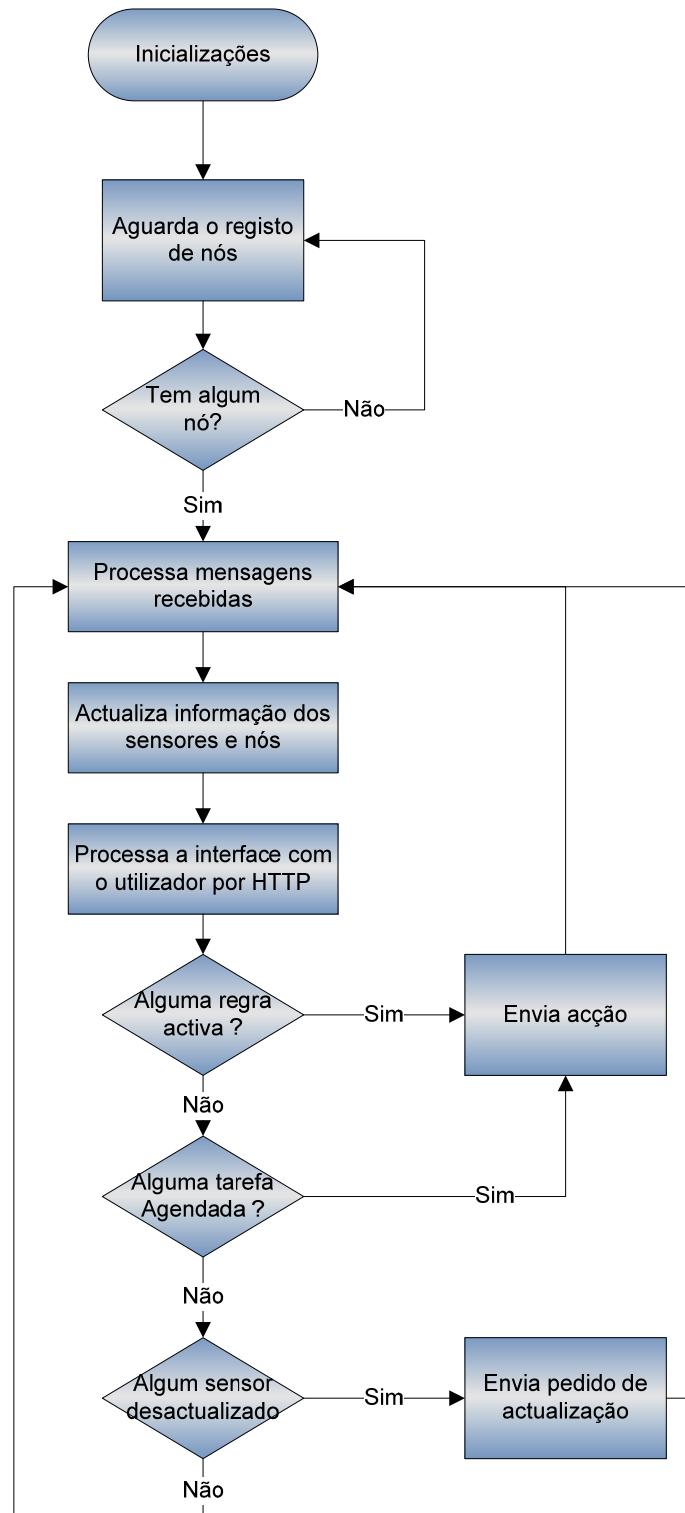


Figura 6.8 – Fluxograma do coordenador



### 6.3.2- Protocolo

De modo a que o sistema funcione com todos os seus elementos de uma forma ordenada foi criado um protocolo. Inicialmente considerou-se a possibilidade de utilizar um protocolo já existente como o ZigBee ou o MiWi. Esta possibilidade foi descartada uma vez que o ZigBee é um protocolo que exige bastantes recursos e introduz muita complexidade ao sistema. Já o MiWi seria uma possibilidade mais apropriada, mas acabou por não ser utilizado, pois a topologia de rede escolhida foi a de estrela, não sendo necessário introduzir a complexidade dos algoritmos de encaminhamento, nem a definição de nós terminais ou encaminhadores. Ao utilizar um protocolo próprio garante-se um controlo total sobre as interações dos vários elementos do sistema, os formatos e os tamanhos das mensagens/tramas.

Por sua vez, as camadas mais baixas, a MAC e PHY, seguem a norma IEEE 802.15.4 garantindo assim a compatibilidade com todos os sistemas que também a sigam. Estas camadas são garantidas através do transceptor fornecido com a PICtail RF que está certificado para esta norma.

Este protocolo atribui ao coordenador o papel principal (*master*) de quase toda a acção. Para este protocolo foram definidas cinco tipos de mensagem:

**HELLO** – mensagem enviada pelos nós quando estes procuram um coordenador de modo a registarem-se na rede. Esta mensagem é enviada no início do funcionamento de cada nó de modo a ganharem um ID de rede. Este processo é algo semelhante ao do DHCP em que um elemento, ao entrar numa rede, pede a atribuição de um IP;

**ACKC** – esta mensagem (ACKC - *Acknowledge Coordinator*) é enviada pelo coordenador ao nó atribuindo-lhe um ID de rede, após recepção de uma mensagem do tipo HELLO. Ao receber esta mensagem o nó sabe que a partir daquele momento já faz parte de uma rede e conhece o seu ID;

**REQ** – esta mensagem (REQ - *Request*) é enviada pelo coordenador ao nó de modo a pedir informação actualizada dos seus sensores.

**ACKN** – mensagem (ACKN - *Acknowledge Node*) enviada pelos nós após a recepção de uma mensagem REQ ou ACT. Esta mensagem serve para confirmação de recepção da mensagem enviada pelo coordenador, mas ao mesmo tempo transporta a informação do estado dos sensores.

**ACT** – esta mensagem (ACT - *Action*) é enviada pelo coordenador a um nó, ordenando que um dos seus actuadores seja accionado.

Para compor as mensagens são utilizados os seguintes campos:

<b>MT</b>	- Message Type	→ Tipo de Mensagem
<b>NT</b>	- Node Type	→ Tipo de Nó
<b>TID</b>	- Transaction ID	→ Identificação da Transacção
<b>ID</b>	- Identifier	→ Número de Identificação
<b>DT</b>	- Device Type	→ Tipo de Dispositivo
<b>VL</b>	- Value	→ Valor

Em seguida, apresenta-se o formato das mensagens especificadas neste protocolo, assim como o seu sentido.

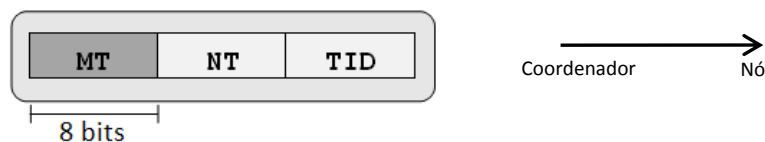


Figura 6.9 – Mensagem HELLO



Figura 6.10 – Mensagem ACKC

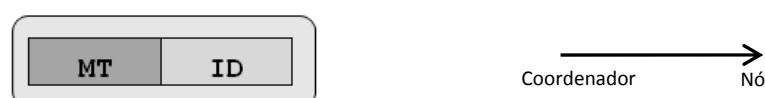


Figura 6.11 – Mensagem REQ

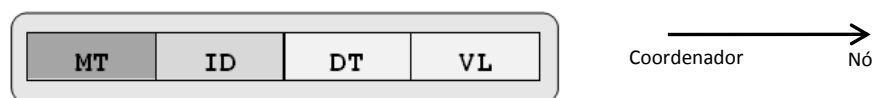


Figura 6.12 – Mensagem ACT

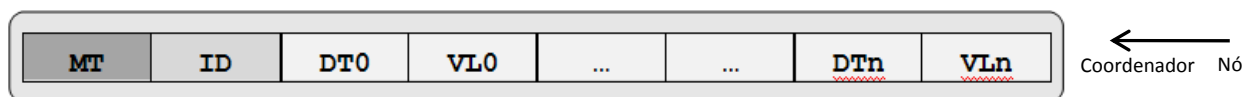


Figura 6.13 – Mensagem ACKN

De referir que cada quadrícula na estrutura das mensagens apresentada representa oito bits (um byte), como se pode ver na Figura 6.9. Todas as mensagens têm no seu primeiro byte o tipo de mensagem de modo a facilitar a leitura. Assim quando se recebe uma mensagem, avaliando o primeiro byte, sabe-se logo de que mensagem se trata, qual a quantidade de dados a ler e como interpretar cada campo da trama.

O protocolo especificado tem o seguinte comportamento. Inicialmente o coordenador fica a aguardar a recepção de alguma mensagem HELLO. Quando um nó é iniciado e vai enviar uma mensagem HELLO ele transmite o tipo de nó (NT) e a identificação da transacção (TID), como mostrado na Figura 6.9. O envio do NT vai permitir ao coordenador saber quais os sensores e actuadores que este elemento dispõe. O TID é um número pseudo-aleatório gerado na altura do envio, de modo a identificar aquela transacção. Isto por que nesta fase os nós não têm ID, e se dois enviarem uma mensagem HELLO na mesma altura, podem surgir problemas de não se saber para quem é a resposta do coordenador e ficarem com o mesmo ID, ocorrendo assim conflito.

Quando o coordenador recebe um HELLO, verifica a NT, adiciona à lista o novo nó e irá começar a monitorizar os sensores associados a esse nó. Ao adicionar o nó, o coordenador, verifica qual o próximo ID livre e vai enviar uma mensagem de ACKC para atribuir este novo ID ao nó. No envio do ACKC, apresentado na Figura 6.10, o TID é o mesmo número recebido no HELLO de modo a que se saiba que esta é uma resposta a um pedido específico.

O nó ao receber o ACKC sabe que entrou na rede e sabe ainda o ID que lhe foi atribuído, ficando agora à espera de novas indicações do coordenador. A partir deste

momento o nó passa a um comportamento meramente reactivo às ordens do coordenador (do ponto de vista de rede, porque de resto continua a efectuar leituras nos seus sensores) na medida em que nunca mais inicia comunicações, apenas envia ACKNs como resposta aos REQ e ACT.

Com o(s) nó(s) registados, o coordenador, verifica há quanto tempo recebeu informação sobre cada nó, e quando passar de um limite predefinido envia um REQ para que o nó envie a informação actualizada. Na mensagem REQ apenas é enviado o ID de modo a identificar qual o nó questionado, como apresentado na Figura 6.11.

A mensagem ACKN tem dupla utilidade, é uma confirmação da recepção das mensagens enviadas pelo coordenador e ao mesmo tempo transmite a informação actualizada dos seus sensores. Deste modo, o número de transmissões efectuadas será menor, fazendo com que o transceptor esteja menos tempo activo, resultando numa maior durabilidade das baterias.

O facto de o ACKN ter um tamanho variável deve-se ao tipo de nó que o transmite. Como se pode ver na Figura 6.13, a trama ACKN é composta pelo tipo de mensagem (MT), o ID do nó destinatário e por pares de tipo de dispositivo (DT) e valor (VL). Estes pares de DT e VL representam cada sensor que o nó tem e o seu valor. Deste modo, se um nó contém dois sensores, por exemplo, um de temperatura e outro de luminosidade, ele enviará uma trama composta por MT, ID e dois pares de DT e VL, sendo chamada de ACKN2. É claro que o MT é diferente para cada variação do ACKN de modo a que o coordenador saiba quantos dados deve ler. O protocolo contempla nós com até nove sensores, sendo assim no máximo um ACKN9.

Quando por algum motivo, o coordenador pretende accionar um actuador, ele envia um ACT que é composto, mais uma vez, com um par de DT e VL, como apresentado na Figura 6.12. Neste caso o DT define qual o dispositivo do nó que o coordenador quer accionar e o VL qual a acção, como por exemplo ligar o AC (DT) na temperatura 25° (VL) do nó 2 (ID). Após o nó receber o ACT deverá executar a ordem recebida e enviar de novo um ACKN.

O comportamento descrito pode ser mais facilmente compreendido através da Figura 6.2 apresentada anteriormente.

Uma vez explicado o protocolo implementado, descreve-se agora o módulo ACOM. Este implementa a camada aplicação deste protocolo. Como toda a gestão da rede é

feita no coordenador, é daqui que todas as interações na rede são iniciadas (à excepção da mensagem HELLO).

O ACOM é a camada mais acima, sendo suportada pelo NCOM e LCOM. É nesta camada que, através da informação disponibilizada pelos outros módulos, o sistema reage em termos de rede. Assim quando um sensor está desactualizado ou é necessário accionar um actuador são enviadas mensagens aos nós. Ao enviar estas mensagens o ACOM apenas sabe se a mensagem teve sucesso ou não, ou seja, se foi confirmada a recepção da mensagem com ACKN. Ao surgir uma mensagem não confirmada, este aguarda um tempo específico e reenvia de novo. Este processo é repetido outro número predefinido de vezes. Se não receber nenhuma confirmação o nó será removido da rede, assim como os sensores e actuadores a este associados.

É importante salientar que embora para o ACOM a mensagem é reenviada um número específico de vezes, esta é enviada muitas mais vezes. O ACOM, ao encarregar o NCOM de enviar a mensagem, não sabe que por cada vez que é para enviar uma trama, esta é enviada mais um número predefinido de vezes pelo NCOM aquando de não confirmação. Esta implementação pretende garantir que a camada intermédia NCOM encarrega-se de enviar mensagens e aguardar confirmação, reportando o resultado. A maneira como o sistema reage a falhas já é com a camada acima, este apenas tenta enviar, reenvia se for necessário, e informa do resultado.

### 6.3.3- Nó

Para a implementação dos nós em termos de *hardware* utilizaram-se duas placas PICDEM Z com duas PICTail RF. O microcontrolador em que foi implementado é o PIC18F4620. Na Figura 6.14 mostra-se a placa utilizada. Os PICTails RF para os nós são diferentes da do coordenador, pois estes são uma versão mais antiga.

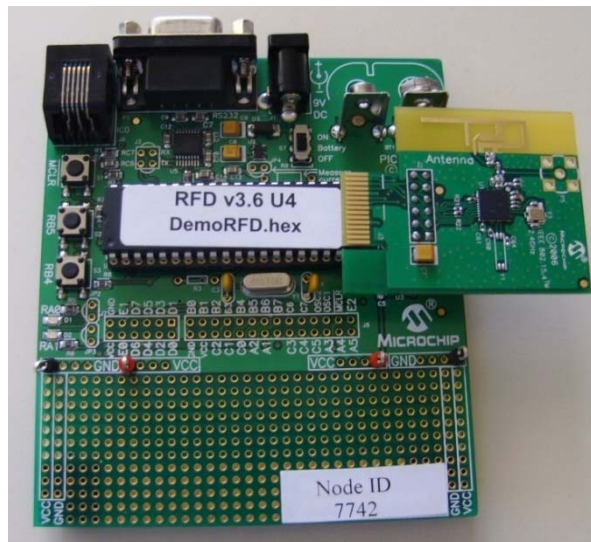


Figura 6.14 – Hardware do nó

Como previamente referido, a modelação dos nós será elaborada através de RdP. Este modelo será único para todos os nós, ou seja, comportamentalmente cada nó será idêntico aos outros. Estes irão variar sim no seu tipo e consequentemente nos sensores e actuadores que dispõem. Deste modo existe apenas uma opção que pode ser por *software*, ou mesmo através de um *jumper*, que irá definir ao sistema que tipo de nó é, sabendo assim quais os seus sensores/actuadores e como os controlar.

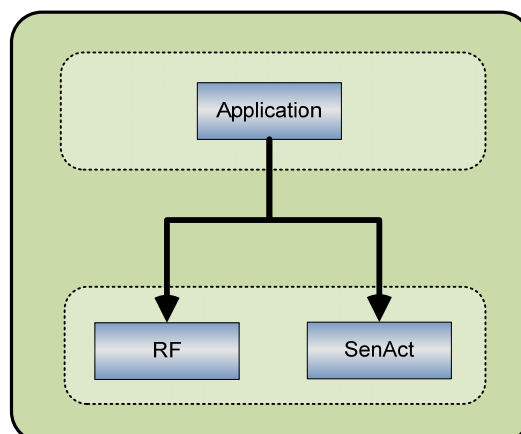


Figura 6.15 – Diagrama de blocos do nó

Em termos de *software* não é necessária uma arquitectura tão complexa como a do coordenador. Nesta arquitectura haverá apenas três módulos. Na Figura 6.15 os módulos mais abaixo são os que interagem directamente com o *hardware*. O módulo *RF* controla o transceptor de modo a enviar e receber mensagens. Já o módulo *SenAct*

é o que vai capturando a informação dos sensores e quando recebe ordens, acciona os seus actuadores. Por sua vez o módulo Application é onde vai estar todo o controlo deste sistema. Este vai consistir numa máquina de estados que implemente o comportamento desejado.

Foi então necessário desenhar/especificar este modelo, inicialmente através de um fluxograma e posteriormente através de RdP.

Na Figura 6.16 apresenta-se o fluxograma que descreve o comportamento desejado.

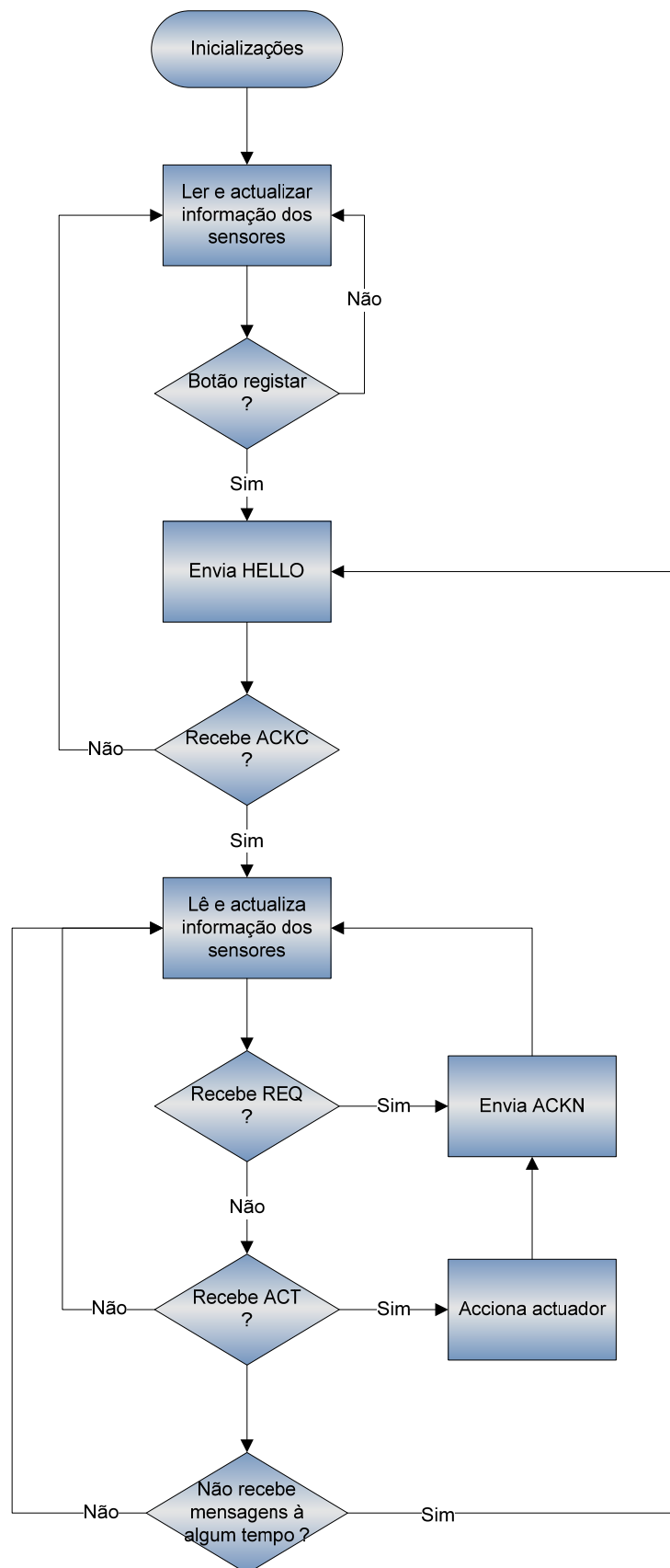


Figura 6.16 – Fluxograma do nó



Com o modelo expresso por fluxogramas passou-se à modelação através das RdP. A RdP associada é apresentada na Figura 6.17.

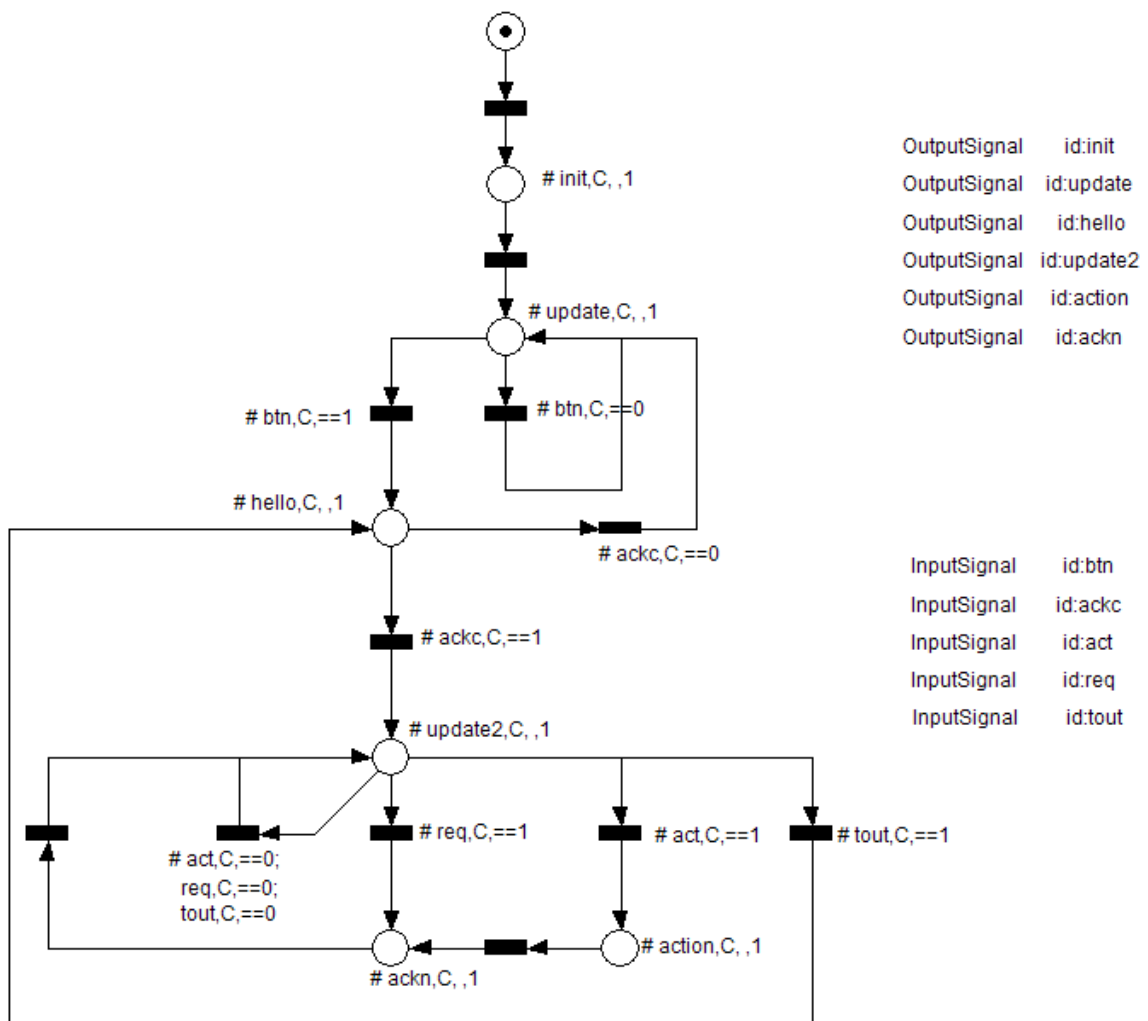


Figura 6.17 – RdP do nó

Uma vez com o modelo expresso por RdP é possível, com as ferramentas descritas anteriormente, a geração do código. Em relação à arquitectura apresentada na Figura 6.15, este modelo comportamental gerado automaticamente corresponderá ao módulo Application. Deste modo, todo o controlo deste sistema será efectuado através do código gerado automaticamente, associado ao modelo RdP apresentado.

Sendo esta uma RdP IOPT é necessário definir os sinais de entrada e saída. Existem sinais de entrada associados às transições e sinais de saída associados a lugares. Neste

sistema a maioria dos sinais de entrada e saída serão relativos ao módulo RF. Estes sinais de entrada (*ackc*, *act*, *req*) surgem por interrupção aquando da chegada de novas mensagens. O sinal de entrada *btn* é uma entrada da rede que tem origem no botão para o registo enquanto que o *tout* é também gerado por interrupção ao fim de um tempo específico sem receber nenhuma mensagem da rede. Já os sinais de saída permitem sinalizar o estado do sistema, assim como determinar a execução de tarefas. Os sinais *ackn* e *hello* quando activos irão indicar ao módulo RF que deve enviar as respectivas mensagens. Já o sinal *init* mostra aos módulos RF e SenAct que o sistema está a arrancar, e que devem efectuar as suas inicializações. Já os sinais de saída *update*, *update2* e *act*, indicam ao módulo SenAct quando deve recolher informação dos sensores, e quando deve accionar um actuador.

O modelo apresentado representa o seguinte comportamento. De início o modelo vai para o lugar que representa a execução das inicializações. Em seguida vai ler os seus sensores e actualizar a informação associada a estes. Depois, se o botão de registo for activado o modelo evolui para o lugar onde a mensagem HELLO será enviada, senão volta ao mesmo lugar. Depois conforme a recepção, ou não, do ACKC segue para o lugar anterior ou avança para outro lugar onde vai ser actualizada de novo a informação dos sensores. A partir deste momento a evolução do modelo será cíclica se o nó não perder comunicação com o coordenador. Daqui o modelo pode seguir quatro caminhos. Se receber um REQ este vai enviar um ACKN e voltar ao lugar anterior; se receber um ACT vai actuar, enviar o ACKN e voltar ao mesmo lugar; se passar um tempo específico sem receber nenhuma mensagem do coordenador vai tentar registar-se de novo na rede. Se o novo registo for bem sucedido volta ao ciclo normal, senão aguarda intervenção do utilizador/instalador. Se nenhum dos três caminhos (transições) anteriores estiver habilitado o modelo permanece no mesmo lugar aguardando.

Uma vez modelada a RdP no SnoopyIOPT criou-se o ficheiro PNML. Este PNML descreve o modelo da RdP e funciona como ponto de interligação para algumas ferramentas, nomeadamente o gerador de código C. Assim utilizou-se a ferramenta PNML2C e gerou-se o código C que implementa a RdP.

Com o modelo da RdP gerado passou-se à integração com os outros módulos. Como este código representa uma RdP IOPT a sua integração numa arquitectura destas é

deveras facilitado. Foi apenas necessário definir como os sinais de entrada e saída do modelo interagem com os outros módulos. Este processo foi facilitado mais uma vez graças à modularização do sistema a implementar. Foi ainda necessário adicionar os bits de configuração do PIC e outras especificidades ao projecto em si.

Posto isto, o sistema estava pronto para se iniciarem os testes.

## **6.4- Teste e Resultados**

De modo a validar o comportamento e o funcionamento do sistema foram efectuados inúmeros testes. Em seguida ir-se-ão apresentar alguns deles. De modo a configurar o sistema foi utilizado o interface desenvolvido através da página *Web*. Um manual de utilização do interface é apresentado em anexo nesta dissertação.

O coordenador foi testado isoladamente na altura do seu desenvolvimento, e só depois foi testado com os nós em conjunto. Os testes isolados ao coordenador são designados testes unitários e serão apresentados em seguida.

### **6.4.1- Testes Unitários**

Com o objectivo de validar o comportamento do coordenador este foi testado isoladamente. Para isso, como referido anteriormente, foi criado uma aplicação em Java que simula os nós, e foi ainda criado um LCOM para Ethernet. Ao validar o comportamento do coordenador por Ethernet, reduz-se a probabilidade de erros no canal de comunicação, garantia que não existe nas RF.

Esta aplicação Java simula dois tipos de nós diferentes, sendo estes seleccionados através de um argumento que é passado pelo ficheiro *batch* que lança a aplicação.

Como estes testes são por UDP sobre Ethernet a cada elemento foi atribuído um IP. Assim ao nó (PC) foi atribuído o IP: 192.168.0.33, e ao coordenador o IP 192.168.0.43. Como não foi possível ter duas aplicações à escuta na mesma máquina e no mesmo porto em simultâneo, foi elaborado um mecanismo para o envio das tramas. Este mecanismo consiste no coordenador enviar as tramas para um porto baseado no ID de quem quer enviar. Assim as mensagens vão para o porto definido por:  $K + ID_{destino}$  (K

é um número constante). Ao lançar as várias aplicações JAVA através do *batch*, este passa como argumento o porto em que cada aplicação se deve por à escuta.

Na Figura 6.18 mostra-se um teste efectuado com apenas um nó. Neste teste observa-se o nó a enviar uma mensagem HELLO, ao qual o coordenador responde atribuindo-lhe um ID. Imediatamente a seguir o coordenador envia um REQ de modo a obter informações sobre os sensores do nó e este responde com ACKN. Pode-se observar que este processo repete-se aproximadamente de cinco em cinco segundos. Se observarmos com cuidado na realidade é de cinco em cinco segundos mais um atraso variável. Este atraso surge porque o módulo Status utiliza o RTCC para marcar os sensores desactualizados, e como referido anteriormente o RTCC actualiza o tempo por *polling* e não por interrupção, resultando sempre em ligeiras variações temporais. Outro atraso introduzido neste mecanismo é que após o Status marcar os nós desactualizados, é o ACOM que posteriormente irá detectar esses nós e enviar um REQ. O tempo que o ACOM demora desde a marcação do Status ao envio do REQ, sofre de ligeiras variações conforme a actividade dos outros módulos do sistema.

Este atraso para as aplicações em questão é completamente irrelevante e é por este motivo que foi implementado deste modo, pois se fosse um factor crítico seria baseado em interrupções.

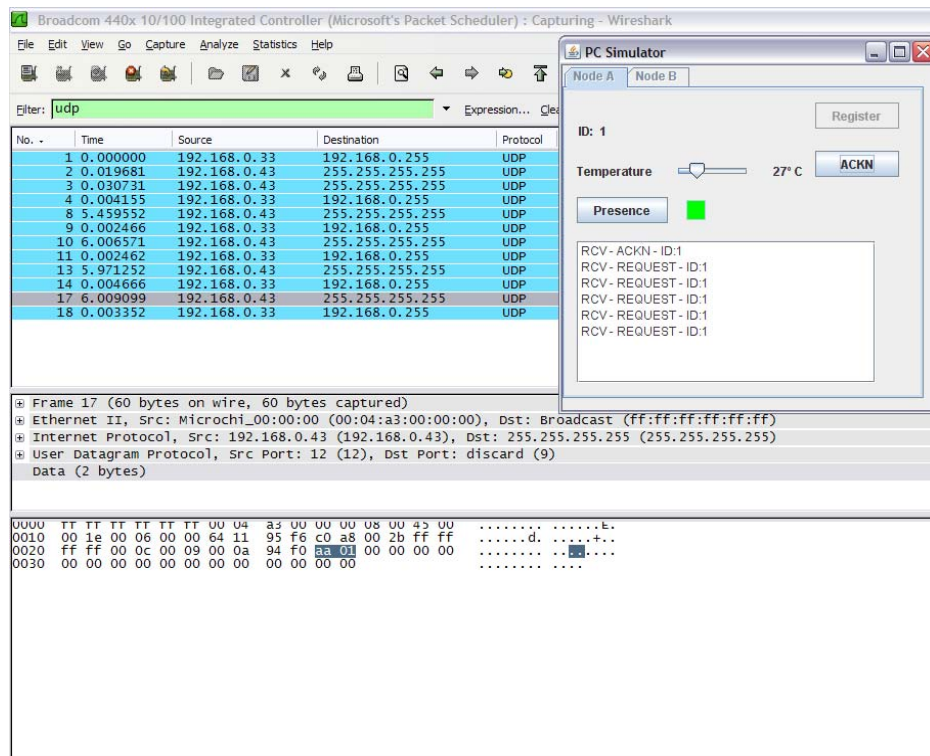


Figura 6.18 – 1º Teste unitário

É na detecção destes pequenos detalhes que o teste unitário tem a sua importância, pois permite analisar os resultados de sistemas isolados, minimizando as incertezas e factores variáveis.

Outro teste apresentado é o seguinte. Foram registados dois nós de tipos diferentes (as aplicações JAVA têm uma apresentação ligeiramente diferente). De modo a simular a introdução de uma regra, foi adicionado código para introduzir a regra ao fim de uns segundos quando os nós já se encontram registados. Esta regra define uma acção para quando o sensor de temperatura do passar os 30° no nó com ID 1, e um proacção quando o valor volta abaixo desse limite. Assim após o registo dos dois nós foi variado o valor da temperatura para cima dos 30 e depois para baixo. Como se pode verificar na Figura 6.19, o nó um após ter enviado um ACKN com o valor acima do limite recebeu uma acção com o valor 0. Em seguida variou-se o valor da temperatura, e quando o nó respondeu ao próximo REQ com ACKN, recebeu a proacção com o valor 2. Neste teste pode-se verificar que o segundo nó continuou o seu funcionamento normal independentemente da actividade do outro.

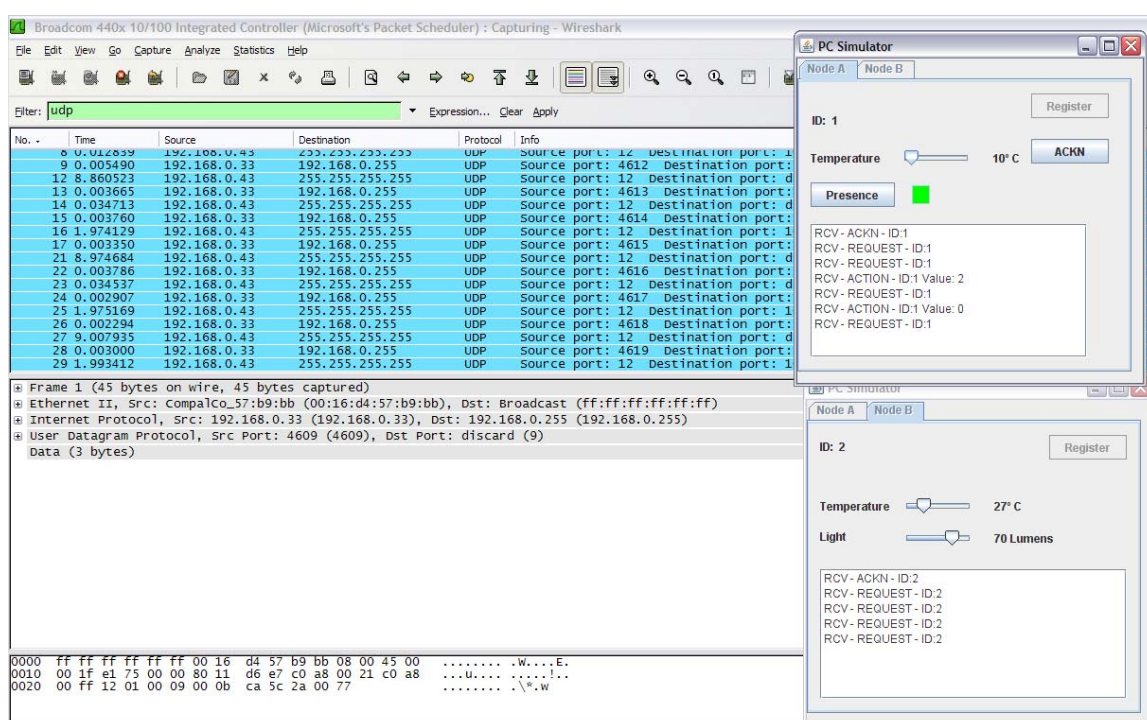


Figura 6.19 – 2º Teste unitário

Muitos outros cenários foram testados de modo a avaliar as funcionalidades implementadas, de forma a garantir o bom funcionamento do sistema. Claro que neste processo foram detectados alguns problemas que foram sendo corrigidos à medida que surgiam.

#### 6.4.2- Testes de Sistema

Nestes testes todo o sistema foi avaliado com o coordenador e os vários nós em simultâneo. Uma vez validado o desempenho do coordenador com os testes unitários, a avaliação do desempenho do sistema foi facilitada pois o comportamento dos nós é deveras simples. Estes testes vêm ainda fornecer as garantias que o sistema funciona utilizando a RF como meio de comunicação.

Nestes não foram utilizados sensores reais, os valores dos sensores eram apenas variados manualmente de modo a verificar a reacção do sistema. Nos testes anteriores essa variação era feita através da aplicação desenvolvida. Agora para variar os valores dos sensores foi utilizada a porta série com comunicações RS232, recebendo assim os valores de cada sensor. De modo a tornar este processo o mais versátil possível foi utilizado o comando por IrDA mencionado anteriormente. O receptor IrDA ao receber os caracteres ASCII (bytes), reenvia por RS232 de modo a dar entrada das variáveis no sistema.

Nos testes apresentados em seguida há que salientar que o analisador de rede (*sniffer*) utilizado está optimizado para o protocolo MiWi ou ZigBee, pelo que existem mensagens que são interpretadas como mensagens destes protocolos, enquanto outras apenas como dados inválidos. No teste apresentado na Figura 6.20 foi simulada a situação do teste anterior mas apenas com um nó. É possível ver a sequência de mensagens desde o registo até à proacção, passando pela acção. Destacam-se os valores do sensor presentes nos ACKN que fazem desencadear as acções. Inicialmente a temperatura encontra-se a 25° estando abaixo do limite (trama 4). Quando o coordenador recebe um ACKN informando que a temperatura se encontra a 66° (trama 6) este reage enviando uma acção, pois a temperatura está acima do limite dos 30°. Quando ao fim de um tempo a temperatura volta novamente abaixo do limite (trama 12) é enviada nova acção (a proacção) ao nó.

ZENA(TM) Packet Sniffer - MiWi(TM) Protocol									
Frame	Time(us)	Len	MAC Frame Control				Seq Num	Encrypted Data	FCS
00001	+31875280 =31875280	5	Type	Sec	Pend	ACK	IPAN		RSSI Corr CRC
			ACK	Y	N	Y	N	0x2D	-10 0x6B OK
} HELLO									
Frame	Time(us)	Len	Invalid Data						FCS
00002	+26096 =31901376	5	0xAC	0x01					RSSI Corr CRC
			0x2D						-06 0x6C OK
} → ACKC									
Frame	Time(us)	Len	Invalid Data						FCS
00003	+3018016 =34919392	4	0xAA						RSSI Corr CRC
			0x01						-08 0x6A OK
} → REQ									
Frame	Time(us)	Len	Invalid Data						FCS
00004	+70224 =34989616	8	0x4C	0x01	0x00				RSSI Corr CRC
			0x19	0x03	0x00				-09 0x6A OK
} → ACKN									
Frame	Time(us)	Len	Invalid Data						FCS
00005	+8666896 =43656512	4	0xAA						RSSI Corr CRC
			0x01						-08 0x6A OK
} → REQ									
Frame	Time(us)	Len	Invalid Data						FCS
00006	+71488 =43728000	8	0x4C	0x01	0x00				RSSI Corr CRC
			0x42	0x03	0x00				-09 0x6A OK
} → ACKN									
Frame	Time(us)	Len	Invalid Data						FCS
00007	+33728 =43761728	6	0x95	0x01					RSSI Corr CRC
			0x7B	0x02					-07 0x6B OK
} → ACT									
Frame	Time(us)	Len	Invalid Data						FCS
00008	+93520 =43855248	8	0x4C	0x01	0x00				RSSI Corr CRC
			0x42	0x03	0x00				-09 0x6A OK
} → ACKN									
Frame	Time(us)	Len	Invalid Data						FCS
00009	+8790352 =52645600	4	0xAA						RSSI Corr CRC
			0x01						-08 0x6C OK
} → REQ									
Frame	Time(us)	Len	Invalid Data						FCS
00010	+71168 =52716768	8	0x4C	0x01	0x00				RSSI Corr CRC
			0x42	0x03	0x00				-12 0x6B OK
} → ACKN									
Frame	Time(us)	Len	Invalid Data						FCS
00011	+8920272 =61637040	4	0xAA						RSSI Corr CRC
			0x01						-06 0x69 OK
} → REQ									
Frame	Time(us)	Len	Invalid Data						FCS
00012	+71504 =61708544	8	0x4C	0x01	0x00				RSSI Corr CRC
			0x00	0x03	0x00				-09 0x6A OK
} → ACKN									
Frame	Time(us)	Len	Invalid Data						FCS
00013	+36560 =61745104	6	0x95	0x01					RSSI Corr CRC
			0x7B	0x00					-06 0x6B OK
} → ACT									
Frame	Time(us)	Len	Invalid Data						FCS
00014	+92528 =61837632	8	0x4C	0x01	0x00				RSSI Corr CRC
			0x00	0x03	0x00				-09 0x6A OK
} → ACKN									

Figura 6.20 – 1º Teste de sistema

A Figura 6.21 mostra outro teste. Neste são registados dois nós e em seguida desligou-se o nó com ID 1. Assim é possível verificar que o coordenador vai tentar, por um certo número de vezes, obter resposta do nó em falha (tramas 9, 12, 13, 14, 17, 18, 19) até que este responda, ou então será removido da rede. Neste processo o coordenador mantém o outro nó actualizado (tramas 7, 10, 15 e 20), demonstrando que não há qualquer interferência para os outros nós da rede.

ZENA(TM) Packet Sniffer - MiWi(TM) Protocol							
Frame	Time(us)	Len	MAC Frame	Seq	Encrypted Data	FCS	
00001	+20450880 =20450880	5	Control 0x002A	Num 0x65		RSSI Corr CRC 0xFC 0x6A 1	} HELLO1
00002	+23760 =20474640	5	Invalid Data 0xAC 0x01 0x65		FCS RSSI Corr CRC 0x00 0x6B 1		} ACKC1
00003	+3019104 =23493744	4	Invalid Data 0xAA 0x01		FCS RSSI Corr CRC 0x00 0x68 1		} REQ1
00004	+71184 =23564928	8	Invalid Data 0x4C 0x01 0x00 0x19 0x03 0x00		FCS RSSI Corr CRC 0xFC 0x6B 1		} ACKN1
00005	+1703024 =25267952	5	Control 0x002A	Num 0x1D		RSSI Corr CRC 0xF6 0x6A 1	} HELLO2
00006	+22336 =25290288	5	Invalid Data 0xAC 0x02 0x1D		FCS RSSI Corr CRC 0x00 0x6A 1		} ACKC2
00007	+3006416 =28296704	4	Invalid Data 0xAA 0x02		FCS RSSI Corr CRC 0xFE 0x69 1		} REQ2
00008	+69568 =28366272	8	Invalid Data 0x4C 0x02 0x00 0x19 0x03 0x00		FCS RSSI Corr CRC 0xF9 0x6B 1		} ACKN2
00009	+8841392 =37207664	4	Invalid Data 0xAA 0x01		FCS RSSI Corr CRC 0xFF 0x6B 1		} REQ1
00010	+2998816 =40206480	4	Invalid Data 0xAA 0x02		FCS RSSI Corr CRC 0xFF 0x6A 1		} REQ2
00011	+71504 =40277984	8	Invalid Data 0x4C 0x02 0x00 0x19 0x03 0x00		FCS RSSI Corr CRC 0xF8 0x6A 1		} ACKN2
00012	+3034224 =43312208	4	Invalid Data 0xAA 0x01		FCS RSSI Corr CRC 0x00 0x6A 1		} REQ1
00013	+3001744 =46313952	4	Invalid Data 0xAA 0x01		FCS RSSI Corr CRC 0xFF 0x69 1		} REQ1
00014	+2997152 =49311104	4	Invalid Data 0xAA 0x01		FCS RSSI Corr CRC 0xFF 0x69 1		} REQ1
00015	+3002720 =52313824	4	Invalid Data 0xAA 0x02		FCS RSSI Corr CRC 0xFF 0x69 1		} REQ2
00016	+71488 =52385312	8	Invalid Data 0x4C 0x02 0x00 0x19 0x03 0x00		FCS RSSI Corr CRC 0xFB 0x6B 1		} ACKN2
00017	+3038448 =55423760	4	Invalid Data 0xAA 0x01		FCS RSSI Corr CRC 0xFF 0x68 1		} REQ1
00018	+2999200 =58422960	4	Invalid Data 0xAA 0x01		FCS RSSI Corr CRC 0xFF 0x6A 1		} REQ1
00019	+2998576 =61421536	4	Invalid Data 0xAA 0x01		FCS RSSI Corr CRC 0x00 0x68 1		} REQ1
00020	+6045408 =67466944	4	Invalid Data 0xAA 0x02		FCS RSSI Corr CRC 0x00 0x69 1		} REQ2
00021	+69888 =67536832	8	Invalid Data 0x4C 0x02 0x00 0x19 0x03 0x00		FCS RSSI Corr CRC 0xFB 0x6A 1		} ACKN2

Figura 6.21 – 2º Teste de sistema



Outro teste efectuado, que não se pode apresentar, foi o de longa duração. Neste registaram-se os dois nós na rede e depois deixou-se o sistema ligado durante dois dias. Ao fim deste tempo verificou-se que ambos os nós continuavam na rede, e a informação destes nós continuava actualizada.

Outros testes foram efectuados com o decorrer do desenvolvimento e na fase final de forma a garantir que todo o sistema funcione de acordo com as especificações e requisitos.

#### **6.4.3- Resultados da ferramenta PNML2C**

Em relação à ferramenta PNML2C os resultados obtidos indicam que esta poderá tornar-se numa ferramenta útil. Em termos específicos, a utilização desta geração automática de código introduz um acréscimo no tamanho do programa e um acréscimo no tempo de execução. Este acréscimo está directamente relacionado com o incremento de linhas de código/instruções que o processador tem que executar. Neste trabalho não foram efectuadas medições específicas relativas ao incremento do tamanho do código, nem a tempos de execução. À data da escrita desta dissertação, outro trabalho em curso pretende elaborar mais sobre este tema.

De qualquer modo o incremento no tamanho do programa é claro quando comparando com uma implementação directa e não tão estruturada (*hard coded*). Como tem vindo a ser defendido nesta dissertação a programação estruturada e modular confere grandes vantagens ao projecto e concretização do mesmo. É por este motivo que este acréscimo do programa e alguma perda de desempenho não é considerada como uma desvantagem evidente.

Claro que existem sistemas onde uma possível perca no desempenho seja crítica e até mesmo casos onde o acréscimo no tamanho do programa é incomportável na plataforma em questão. Muitas vezes estas situações têm mesmo que ser resolvidas através da implementação directa em Assembly.

Este acréscimo de código ao projecto não é tão significativo quando apenas são feitos acrescentos ao modelo inicial. Isto deve-se ao facto de a ferramenta implementar uma arquitectura baseada nas características das RdP IOPT. Esta arquitectura encontra-se apresentada na Figura 5.2, e é sempre implementada quer a RdP tenha oito ou oitenta

lugares. Deste modo existe sempre um acréscimo fixo no uso da ferramenta, mas a partir daí o acréscimo é gradual e está directamente relacionado com o tamanho da rede em si.

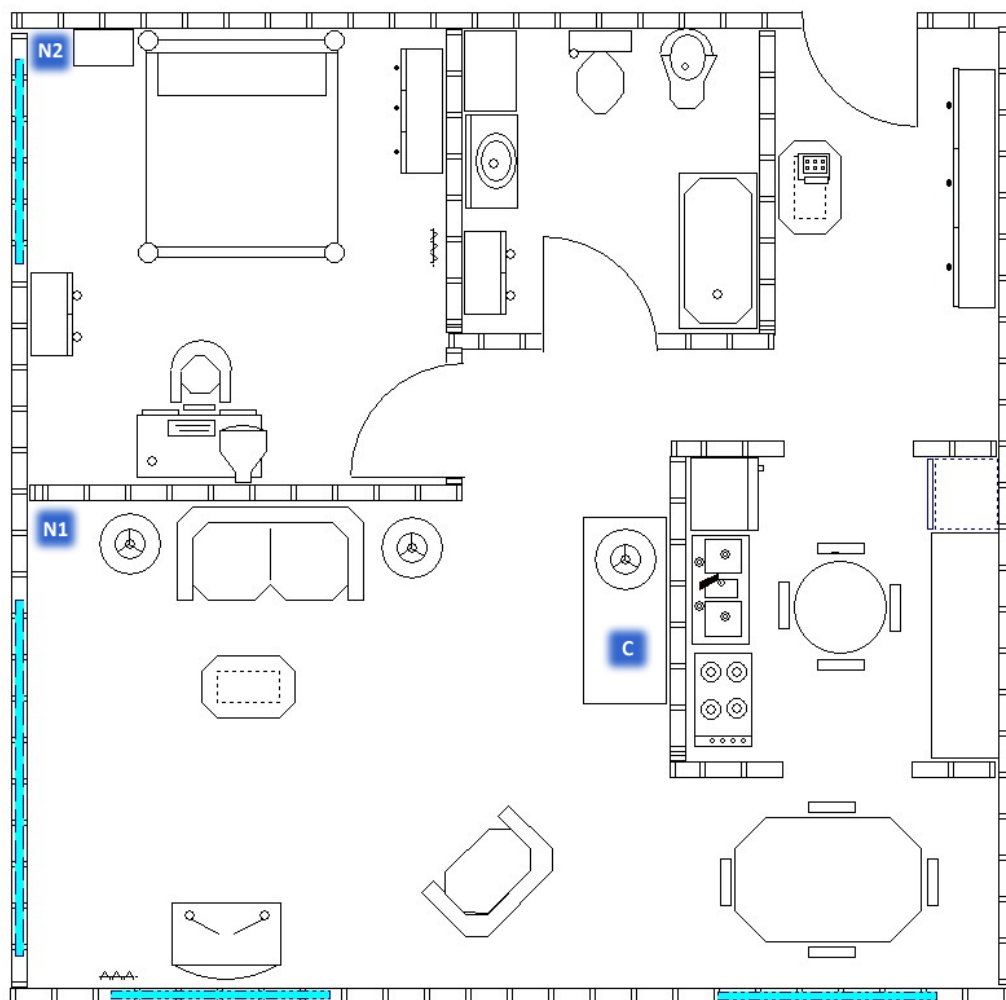
Em termos de esforço para quem desenvolve o sistema, a ferramenta apresenta-se como uma vantagem. De certo modo força a especificação do sistema através de um modelo, o que é sempre vantajoso quando comparando com implementações que não são devidamente especificadas e modeladas. Por outro lado ao forçar a modelação, esta garante que a partir desse ponto o esforço de concretização será minimizado. Garante ainda que a transição do modelo para sua implementação é feita de uma forma fiel e precisa.

Do ponto de vista desta dissertação a ferramenta PNML2C forneceu grande dinamismo ao desenvolvimento dos nós. Foi possível colocar a interagir dois sistemas, o coordenador e os nós, desenvolvidos através de metodologias diferentes. O código gerado foi capaz de implementar o sistema pretendido sem que fossem necessárias grandes modificações ao código.

#### **6.4.4- Modelo de Aplicação**

De modo a simular e demonstrar a aplicabilidade do sistema foi desenhado uma modelo de aplicação. Este modelo aplica a rede de sensores-actuadores desenvolvida a uma casa de modo a conferir-lhe algumas funcionalidades de domótica. Começou-se então por projectar uma casa simples no qual seriam instalados dois nós em divisões diferentes, mais especificamente no quarto e na sala. Na Figura 6.22 mostra-se a localização dos dois nós (N1 e N2) e do coordenador (C). Esta rede tem apenas dois nós devido ao facto de ser este o número de placas disponível para este trabalho.

Com esta topologia foram então especificados dois tipos de nós diferentes. O tipo B, correspondente ao nó2, é equipado com sensores de temperatura e luminosidade, como actuadores pode controlar os estores, a iluminação e o AC da divisão. Já os nós tipo A dispõem do mesmo que os do tipo B mas acresce um sensor de presença, a possibilidade de enviar comandos infravermelhos para a TV e outros aparelho e ainda activação de um alarme.



**Figura 6.22** – Planta do modelo de aplicação

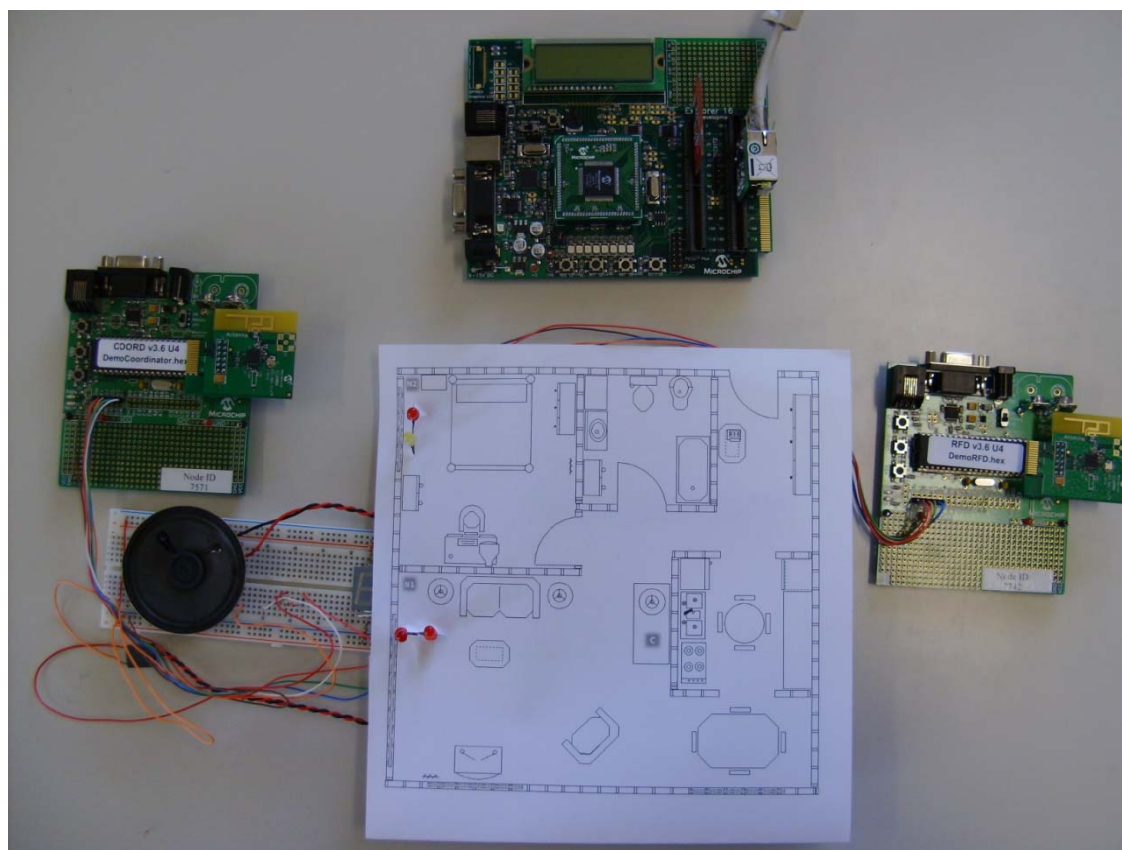
Na Figura 6.23 apresenta-se ainda o modelo 3D desta casa, onde podemos observar os dois nós pendurados na parede do quarto e da sala, enquanto que o coordenador é colocado em cima de uma mesa de forma a estar conectado à Internet.

Uma vez bem definido este modelo de aplicação, foram efectuados alguns ensaios que replicaram este esquema específico.



**Figura 6.23** – Esquema 3D do modelo de aplicação

De modo a testar/ensaiar este modelo de aplicação foi montada um protótipo real desta casa de forma a observar a evolução da rede. Nesta montagem os sensores e actuadores não são reais, apenas demonstram a ideia e o conceito através de entradas e saídas visíveis do sistema. Este protótipo é apresentado na Figura 6.24.



**Figura 6.24** – Protótipo do modelo de aplicação

Numa fase mais avançada da demonstração foi montado um outro protótipo, desta vez com actuadores reais. Estes são uma ventoinha, duas luzes, um alarme sonoro e um *display* de oito segmentos para mostrar a configuração actual do AC. Já os sensores continuaram a ser simulados e a variação dos seus valores é efectuada através do comando infravermelhos que comunica com as placas.



**Figura 6.25** – Segundo Protótipo

## 7- Conclusões

Com a conclusão deste trabalho verifica-se que é possível a implementação um sistema aplicado à área da domótica, totalmente implementado com tecnologias de baixo custo. Todo este sistema com funcionalidades avançadas foi implementado com tecnologias extremamente baratas e com desempenhos bastante competitivos.

Em seguida, na Tabela 7.1, apresentam-se os custos dos materiais utilizados (*Bill Of Materials* - BOM) para este trabalho. O custo total apresentado na tabela é algo significativo se for analisado como o preço de um sistema deste género. Mas é necessário ter em conta que foram utilizadas placas e ferramentas próprias para o desenvolvimento e não para a produção.

<b>Ferramenta</b>	<b>Custo</b>
Ethernet PICTail Plus Daughter Board	39,99 USD
MRF24J40MA PICTail Plus 2.4GHz RF Card	18,95 USD
Explorer 16	129,99 USD
PIC24FJ128GA010 PIM	25,00 USD
PICDEM Z 2.4 GHz DEMO KIT	269,99 USD
<b>Total USD</b>	483,92 USD
<b>Total EUR</b>	383,83 EUR

**Tabela 7.1** – Custo das ferramentas de desenvolvimento

Por outro lado se analisarmos o preço destes componentes e não das ferramentas de desenvolvimento verifica-se uma enorme redução dos custos, cerca de 84%, conforme a Tabela 7.2. É claro que a este valor acrescem os custos de produção do PCB para cada elemento, assim como os custos de alguns componentes que são necessários. É importante referir ainda que neste balanço utiliza-se a o transceptor montado num módulo certificado. Este pode ser adquirido isoladamente a um preço muito reduzido, mas o esforço e o custo da sua integração com a antena, assim como os da posterior certificação muitas vezes torna o processo desvantajoso.

<b>Componente</b>	<b>Quantidade</b>	<b>Preço Uni.</b>	<b>Sub-Total</b>
PIC24FJ128GA010	1	6,21 USD	6,21 USD
ENC28J60	1	3,03 USD	3,03 USD
MRF24J40MA Radio Transceiver Module	3	18,95 USD	56,85 USD
PIC18F4620	2	5,72 USD	11,44 USD
		<b>Total USD</b>	77,53 USD
		<b>Total EUR</b>	61,49 EUR

**Tabela 7.2** – Custo dos componentes

Tendo em conta o custo dos materiais deste sistema e as suas funcionalidades, considerara-se que esta solução é de muito baixo custo. É claro que este sistema, em comparação com as soluções de domótica apresentadas, não tem qualquer componente multimédia, mas permite todo um automatismo e controlo remoto de uma casa.

Deste modo demonstrou-se o potencial destes recentes componentes de baixo custo, e como estes podem ser aplicadas a uma área como a domótica.

De modo a que esta solução pudesse ser implementada num sistema real seria necessário haver electrodomésticos e dispositivos compatíveis com este sistema. Como a produção de tais equipamentos é descartada de imediato, o passo a dar seria colocar esta solução a interagir com os dispositivos já existentes no mercado que comunicam por mecanismos parecidos.

Por exemplo seria bastante simples o coordenador da rede incluir um *modem* PLC (por exemplo da Yitran) de modo a introduzir os comandos na rede eléctrica num formato conhecido como o X10. Deste modo todos os dispositivos deste género receberiam os comandos. Pode-se ainda adicionar a *Stack* ZigBee de forma a saber como interagir com os vários dispositivos existentes no mercado para esta tecnologia.

Há que salientar ainda as vantagens que advieram da correcta modelação e especificação do sistema. Ao se ter criado uma arquitectura modular do sistema foi possível isolar certos problemas que ficam resolvidos e encapsulados dentro de um único módulo. Desta forma existe uma abstracção vantajosa tanto na fase de desenvolvimento como na posterior actualização do sistema.

A modelação dos nós através de RdP apresentou-se muito vantajosa na medida em que permitiu uma especificação completa do modelo comportamental. O uso das RdP em

modelos não muito extensos como este permite uma compreensão comportamental do sistema bastante intuitiva. É assim, portanto, um formalismo de modelação simples mas ao mesmo tempo conciso, que para um sistema deste género parece indicado. Por outro lado a geração automática de código traduz-se numa grande vantagem, pois garante uma correcta interpretação e tradução do modelo, facilitando ainda possíveis alterações que o sistema possa vir a sofrer.

Ficou assim validada a ferramenta PNML2C, que embora ainda não totalmente finalizada, apresentou resultados positivos, deixando a indicação que no futuro poderá vir a ser extremamente útil para a passagem da fase do modelo à fase de implementação, garantindo uma fiel interpretação do modelo.



## Referências

- [Akyildiz – Kasimoglu, 04] I.F. Akyildiz, I.H. Kasimoglu, “Wireless sensor and actor networks: research challenges”, *AdHoc Networks Journal* 2(4), pp. 351 – 367, 2004
- [Anderson-Najm, 03] Jason Anderson, Farid Najm, “Switching Activity Analysis and Pre-Layout Activity Prediction for FPGAs,”, *International Workshop on System-Level Interconnect Prediction, Proceedings of the 2003 international workshop on System-level Interconnect Prediction*, ACM, ISBN: 1-58113-627-7, 2003.
- [Barros, 96] Barros, João Paulo M. P. Ramos, “CpPNeTS: Uma Classe de Redes de Petri de Alto-nível Implementação de um sistema de suporte à sua aplicação e análise”, dissertação apresentada na Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa, como parte dos requisitos necessários à obtenção do grau de Mestre em Engenharia Informática, Lisboa, 1996.
- [Barros-Gomes, 04] João Paulo Barros, Luís Gomes, " Operational PNML: Towards a PNML Support for Model Construction and Modification", *Workshop on the Definition, Implementation and Application of a Standard Interchange Format for Petri Nets*, Satellite event at the International Conference on Application and Theory of Petri Nets 2004, 2004.
- [Benveniste et al, 02] A. Benveniste, P. Caspi, P. Le Guernic, H. Marchand, J.-P. Talpin, and S. Tripakis. “A protocol for loosely time-triggered architectures.”, *Proceedings of EMSOFT’02* A. Sangiovanni-Vincentelli and J. Sifakis, Págs. 252–266, Springer, 2002.
- [Bormann-Cheung, 97] D. S. Bormann, P. Y. Cheung “Asynchronous wrapper for heterogeneous systems.”, *Computer Design: VLSI in Computers and Processors*, IEEE International Conference (ICCD) Proceedings, ISBN: 0-8186-8206-X, 1997.
- [Buck et.al, 94] J. Buck, S. Ha, E.A. Lee, e D.G Messerschmitt, “Ptolemy: A Framework for Simulating and Prototyping Heterogeneous Systems”, *International Journal of Computer Simulation*, Special Issue on Simulation Software Development, Vol. 4 Págs. 155-182, 1994.
- [Chamusca , 06] Alexandre Chamusca, “A inteligência que se instala” *DOMÓTICA & SEGURANÇA ELECTRÓNICA*, Editado pela Ordem dos Engenheiros, ISBN 978-972-98843-8-2, 2006.
- [Chapiro, 84] D. M. Chapiro, “Globally-Asynchronous Locally-Synchronous Systems”, Ph.D. thesis, Stanford University, 1984.

- [Cheong et al, 03] E. Cheong, J. Liebman, J. Liu, and F. Zhao, "TinyGALS: A programming model for event-driven embedded systems", Proceedings of the Eighteenth Annual ACM Symposium on Applied Computing, ACM, Págs. 698–704, 2003.
- [Cheong-Liu, 04] Elaine Cheong, Jie Liu, "galsC: A Language for Event-Driven Embedded Systems", Technical Memorandum UCB/ERL M04/7, Universidade da California, Berkeley, 2004.
- [Conceição, 04] Conceição, Paulo Alexandre Meira da, "De Casos de Uso a Redes de Petri: Uma aplicação à monitorização de edifícios", Dissertação de Mestrado em Engenharia Informática na Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa, 2004.
- [Cortadella et al, 96] Jordi Cortadella, Michael Kishinevsky, Alex Kondratyev, Luciano Lavagno, Alex Yakovlev, "Petrify: a tool for manipulating concurrent specifications and synthesis of asynchronous controllers", XI Conference on Design of Integrated Circuits and Systems, 1996.
- [Dasgupta- Yakovlev, 06] Sohini Dasgupta, Alex Yakovlev, "Modeling and Performance Analysis of GALS Architectures", System-on-Chip, 2006. International Symposium, Págs. 1-4, ISBN: 1-4244-0621-8, 2006.
- [Dasgupta- Yakovlev, 07] S. Dasgupta e A. Yakovlev, "Comparative analysis of GALS clocking schemes", in Computers & Digital Techniques, IET, Vol. 1-2, Págs. 59-69, 2007.
- [Dijkstra, 92] Edsger W. Dijkstra, "On the Economy of Doing Mathematics", Lecture Notes In Computer Science, Proceedings of the Second International Conference on Mathematics of Program Construction, Vol. 669 Springer-Verlag, 1992.
- [Dulman et.al, 06] S. Dulman, S. Chatterjea, P. Havinga, "Introduction to Wireless Sensor Networks", The Industrial Information Technology Handbook, Richard Zurawski, Section V – Networked Embedded Systems, Cap. 31, CRC, ISBN 0849328241, 9780849328244, 2006.
- [Fernández et al, 07] C. Fernández, Rajkumar Raval, C. Bleakley, "GALS SoC Interconnect Bus for Wireless Sensor Network Processor Platforms" in Great Lakes Symposium on VLSI, Proceedings of the 17th ACM Great Lakes symposium on VLSI, Págs. 132-137, ACM, ISBN: 978-1-59593-605-9, 2007.
- [Gomes, et.al, 97] Luís Gomes, Anikó Costa, Carlos Soares, João-Paulo Barros, A. Steiger-Garção, "Campus-Guard: a domot targeted for integrated building monitoring and control", 2nd Int. Conference on Intelligent Buildings, New Methods and Technologies in Planning and Construction of Intelligent Buildings II, Págs. 53-64, 1997

- [Gomes, et al, 98] Luís Gomes, João Paulo Barros, Anikó Costa, "Towards Integrated Modelling of Intelligent Building Systems"; Second European Conference on Product and Process Modelling in the Building Industry; Págs. 19-21, CRC, 1998.
- [Gomes, 97] Gomes, Luís Filipe dos Santos, "Redes de Petri Reactivas e Hierárquicas – Integração de Formalismos no Projecto de Sistema Reactivos de Tempo Real", Tese de Doutoramento em Engenharia Electrotécnica na Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa, 1997.
- [Gomes et. al, 04] Luís Gomes, João Paulo Barros, Rui Pais, "From Non-Autonomous Petri Net Models to Code in Embedded Systems Design", DESDes 04 - 2nd International Workshop on Discrete-Event System Design, 15-17, 2004.
- [Gomes et al, 06] Luís Gomes, João Paulo Barros e Anikó Costa, "Modeling Formalisms for Embedded Systems", The Industrial Information Technology Handbook, Richard Zurawski, Section I, Cap. 5, CRC, ISBN 0849328241, 9780849328244, 2006.
- [Gomes et al, 07] Luís Gomes, João Paulo Barros, Anikó Costa, Ricardo Nunes "The Input-Output Place-Transition Petri Net Class and Associated Tools", Industrial Informatics, 2007 5th IEEE International Conference, Vol. 1, Págs: 509 - 514, 2007
- [Gupta et.al, 06] Sumit Gupta, Hiren Patel, Sandeep Schkla, Rajesh Gupta, "Design Issues for Networked Embedded Systems", The Industrial Information Technology Handbook, Richard Zurawski, Section IV – Networked Embedded Systems, Cap. 29, CRC, ISBN 0849328241, 9780849328244, 2006.
- [Heath et al, 05] Matthew W. Heath, Wayne P. Burleson, Ian G. Harris, Member, "Synchro-Tokens: A Deterministic GALS Methodology for Chip-Level Debug and Test", IEEE Transactions on Computers, Vol. 54-12, Págs. 1532- 1546, 2005.
- [Heidemann-Govindan, 05] John Heidmann, Ramesh Govidan "Embedded Sensor Networks", Handbook of Networked and Embedded Control Systems, Dimitrios Hristu-Varsakelis, Willian Vevine; Section V – Networking, Birkhäuser, ISBN 0817632395, 9780817632397, 2005.
- [Hemani et al, 99] A. Hemani, T. Meincke, S. Kumar, A. Postula, T. Olsson, P. Nilsson, J. Oberg, P. Ellervee, D. Lundqvist "Lowering power consumption in clock by using globally asynchronous, Annual ACM IEEE Design Automation Conference, Proceedings of the 36th ACM/IEEE conference on Design automation, ACM, ISBN:1-58133-109-7, 1999.

- [Heuser, 91] Heuser, Modelação Conceitual de Sistemas. (1a. Edição, EBAI - 1988) 2a. Edição, Campinas: UNICAMP (IV Escola Brasileiro-Argentina de Informática), 1991.
- [Iyer-Marculescu, 02] Anoop Iyer, Diana Marculescu, "Power and Performance Evaluation of GALS", International Symposium on Computer Architecture , Proceedings 29th Annual International Symposium, Págs. 158-168, IEEE, ISBN: 1063-6897, 2002.
- [Jensen, 97] Jensen, Kurt, "A brief introduction to Coloured Petri Nets", Lecture Notes in Computer Science, Tools and Algorithms for the Construction and Analysis of Systems, Págs. 203-208, Springer-Verlag, ISBN:3-540-62790-1, 1997.
- [Jia-Vermuri, 05] X. Jia, R. Vermuri, "Using GALS Architecture to Reduce the Impact of Long Wire Delay on FPGA Performance", Design Automation Conference, 2005. Proceedings of the ASP-DAC 2005, Vol. 2, Págs. 1260-1263, IEEE, ISBN: 0-7803-8737-6, 2005.
- [Jünger et.al, 00] Matthias Jünger, Ekkart Kindler, Michael Weber "The Petri Net Markup Language", Workshop on Algorithms and Tools for Petri Nets, 2000.
- [Körber et.al, 05] Hans-Jörg Körber, Housam Wattar, Gerd Scholl, Wolfgang Heller, "Embedding a Microchip PIC18F452 based commercial platform into TinyOS", 2005.
- [Krstić et al, 05] Miloš Krstić, Eckhard Grass, Christian Stahl, "Request-driven GALS Technique for Wireless Communication System", Asynchronous Circuits and Systems, ASYNC, Proceedings 11th IEEE International Symposium Págs. 76- 85, ISBN: 0-7695-2305-6, 2005.
- [Krstic et al, 06] M. Krstic, C. Grass, e M. Piz, "System integration by request-driven GALS design", Computers and Digital Techniques, IEEE Proceedings, Vol. 153-5, Págs. 362-372, 2006.
- [Lino, 03] Lino, Rui Manuel Gonçalves, "Detecção de Falhas em Sistemas de Automação utilizando Redes de Petri", Dissertação de Mestrado no Departamento de Departamento de Ciências dos Materiais na Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa, 2003.
- [Maldonado, 06] Eduardo Maldonado, "Directiva Europeia 2002/91/CE sobre o Desempenho Energético dos edifícios", INETI – Seminário GREEN-IT, Universidade do Porto, 2006.
- [Masters, 08] Microchip 2008 MASTERS Conference Documentation, Phoenix, Arizona, 2008
- [Meincke et al, 98] Thomas Meincke, Ahmed Hemani, Shashi Kumar, Peeter Ellervee, Johnny ...berg, Dan Lindqvist, Hannu Tenhunen, Adam Postula, "Evaluating benefits of Globally Asynchronous Locally Synchronous VLSI architecture", Proceedings of the 16th Norchip, 1998.

- [Mousavi et al, 04] MohammadReza Mousavi, Paul Le Guernic, Jean-Pierre Talpin, Sandeep Kumar Shukla, Twan Basten "Modeling and Validating Globally Asynchronous Design in Synchronous Frameworks", Design, Automation, and Test, Vol. 1, Págs. 10384, IEEE, ISBN: 0-7695-2085-5-1, 2004.
- [Murata, 89] Murata, Tadao, "Petri Nets: Properties, Analysis and Applications", Proceedings of IEEE, Vol. 77-4, 1989.
- [Muttersbach et al, 00] J. Muttersbach, T. Villiger, W. Fichtner, "Practical Design of Globally-Asynchronous Locally-Synchronous Systems", ASYNC 2000 Proceedings. Sixth International Symposium, Págs. 52-59, IEEE, 2000.
- [Najibi et al, 05] Mehrdad Najibi, Kamran Saleh, Mohsen Naderi, Hossein Pedram, Mehdi Sedighi "Prototyping Globally Asynchronous Locally Synchronous Circuits on Commercial Synchronous FPGAs", Rapid System Prototyping, (RSP 2005), The 16th IEEE International Workshop, Págs. 63- 69, ISBN: 0-7695-2361-7, 2005.
- [Nielsen et al, 01] Moegens Nielson, Vladimiro Sassone, Jiri Srba, "Towards a Notion of Distributed Time for Petri Nets", Applications and Theory of Petri Nets, 22nd International Conference ICATPN, Proceedings, Springer, 2001.
- [Niyogi-Marculescu, 05] K. Niyogi, D. Marculescu, "System Level Power and Performance Modeling of GALS Point-to-point Communication Interfaces", Proceedings of the 2005 international symposium on Low power electronics and design, ACM, 2005.
- [Nunes, 03] Renato Nunes, "DomoBus - A New Approach to Home Automation", 8CLEEE - 8th International Congress on Electrical Engineering, Portugal, Págs. 2.19-2.24, 2003.
- [Nunes, 04b] Renato Nunes, "A Communication Infrastructure for Home Automation", International Conference on Computer, Communications and Control Technologies, Págs. 56-61 2004.
- [Nunes, 05] Renato Nunes, "Implementing Tiny Embedded Systems with Networking Capabilities", IADIS International Conference on Applied Computing, 2005.
- [Nunes, 06] Renato Nunes, "Decentralized Supervision for Home Automation", MELECON 2006 - The 13th IEEE Mediterranean Electrotechnical Conference, Benalmádena, Spain, May 2006.
- [Nunes et al, 07] Ricardo Nunes, Luís Gomes, João Barros, "A Graphical Editor for the Input-Output Place-Transition Petri Net Class", 12th IEEE Conference on Emerging Technologies and Factory Automation, ETFA, Págs. 25-28, 2007.

- [Pais, 04] Pais, Rui Manuel Carvalho, "Geração de Executores e Analisadores de Redes de Petri", Dissertação de Mestrado no Departamento de Engenharia Informática na Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa, 2004.
- [Peeters, 96] A. M. Peeters. "Single-Rail Handshake Circuits" Tese de Doutoramento, Eindhoven University of Technology, ISBN:0-8186-7098-3, 1996.
- [Pender, 03] Tom Pender, "UML Bible", Wiley, ISBN: 978-0764526046, 2003.
- [Peterson, 77] Peterson, James, "Petri Nets", ACM Computing Surveys, Vol. 9-3, Págs. 223-252, 1977.
- [Petri, 62] Carl Adam Petri, Dissertação de doutoramento, "Kommunikation mit Automaten (Communication with Automata)", 1962.
- [Petri, 63] Fundamentals of a Theory of Asynchronous Information Flow. 1<sup>st</sup> IFIP World Computer Congress. Munich 1962, North Holland, pp 386-390, 1963.
- [Petri, 67] "Grundsätzliches zur Beschreibung diskreter Prozesse (Fundamentals on the description of discrete processes)". 3rd Colloquium on Automata Theory in Hannover, Birkhäuser-Verlag, pp121-140, 1967.
- [PIC24F, 07] PIC24F Family Reference Manual, 2007.
- [Reis, 08] Reis, Tiago Miguel Correia, "Partição de modelos de Redes de Petri", Dissertação de Mestrado em Engenharia de Electrotécnica e de Computadores na Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa, 2004.
- [Reisig, 85] "Petri nets: an introduction", Springer-Verlag, ISBN:0-387-13723-8, 1985.
- [Royal-Cheung, 03] Andrew Royal, Peter K. Cheung, "Globally Asynchronous Locally Synchronous FPGA Architectures", Field Programmable Logic and Applications, Lecture Notes in Computer Science (LNCS), Págs: 355-364, Springer-Verlag, 2003.
- [Rumbaugh et al, 99] James Rumbaugh, Ivar Jacobson, Grady Booch, "The Unified Modeling Language Reference Manual", Addison-Wesley Professional, ISBN: 0-201-30998-X, 1999.
- [Sgroi, 00] M. Sgroi, L. Lavagno e A. Sangiovanni-Vincentelli, "Formal Models for Embedded Systems Design", IEEE Design and Test of Computers, Vol. 17, Págs. 14-17, 2000.
- [Shemie, 97] Shemie, Sam; "Communications, Control & Information Technology Systems in Intelligent Buildings"; in New Methods and Technologies in Planning and Construction of Intelligent Buildings II, Proceedings of the 2<sup>nd</sup> IB/IC Intelligent Buildings Congress, 1997.

- [Shin et al, 07] Soo Young Shin; Hong Seong Park; Sunghyun Choi; Wook Hyun Kwon, "Packet Error Rate Analysis of ZigBee Under WLAN and Bluetooth Interferences", *Wireless Communications, IEEE Transactions*, Vol6-8, Págs. 2825–2830, 2007.
- [Singh-Theobald, 03] Montek Singh, Michael Theobald, "Generalized Latency-Insensitive Systems for GALS Architectures" *Proceedings of FMGALS'03*, 2003.
- [Sparso- Furber, 02] Jens Sparso, Steve Furber, "Principles of Asynchronous Circuit Design – A System Perspective", *Kluwer Academic Publishers*, 2002.
- [Stahl et al, 05] Christian Stahl, Wolfgang Reisig, Milos Krstić "Hazard Detection in a GALS Wrapper: a Case study", *Application of Concurrency to System Design, ACSD 2005, 5th International Conference*, Págs. 234- 243, ISBN: 0-7695-2363-3, 2005.
- [Taylor et al, 00] G. Taylor, S. Moore, S. Wilcox, e P. Robinson. "An onchip dynamically recalibrated delay line for embedded selftimed systems", *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems*, ISBN:0-7695-0586-4, 2000.
- [Varchola-Drutaroský, 07] Michal Varchola, Miloš Drutaroský, "ZigBee based Home Automation Wireless Sensor Network", *Acta Electrotehnica et Informatica*, Vol.7-4, WSEAS, 2007.
- [Weber-Kindler, 02] Michael Weber, Ekkard Kindler "The Petri Net Markup Language", *LNCS Petri Net Technology for Communication-Based Systems*, ISBN 978-3-540-20538-8, 2002.
- [Wegrzyn et al, 97] Marek Wegrzyn, Pawel Wolanski, Marian Adamski, Joao L. Monteiro, "Coloured Petri Net Model of Application Specific Logic Controller Programs", *Industrial Electronics, ISIE '97, Proceedings of the IEEE International Symposium*, Vol. 1, 7-11 p.158 - 163, 1997.
- [Yakovlev, 02] Alex Yakovlev "Is the die cast for the token game?", *Lecture Notes In Computer Science, Proceedings of the 23rd International Conference on Applications and Theory of Petri Nets, ICATPN*, Vol. 2360, Págs. 70-79, 2002.
- [Yun-Donohue, 96] Kenneth Y. Yun, Ryan P. Donohue "Pausible clocking: A first step toward heterogeneous systems.", *Proc. International Conf. Computer Design (ICCD)*, Págs. 158 - 168 1996.
- [Yun-Dooply, 99] Kenneth Y. Yun, Ayoob E. Dooply, "Pausible clocking based heterogeneous systems", *IEEE Transactions on VLSI Systems*, Vol. 7-4, Págs. 482–488, 1999.
- [Zurawski et al, 94] Zurawski, R.; MengChu Zhou, "Petri nets and industrial applications: A tutorial", *Industrial Electronics, IEEE Transactions*, Vol. 41, Págs. 567-583, 1994.

## Referências Eletrónicas

- [Brauer-Reisig, 06] W. Brauer e W. Reisig, “Carl Adam Petri and ‘Petri Nets’”, 2006.  
<http://www.informatik.uni-hamburg.de/TGI/PetriNets/history/>, Acedido em Janeiro 2009.
- [Bystrov-Shang, 06] Alex Bystrov, Delong Shang, “Predictive Synchronisation in the new Generation of GALS”, School of Electrical Electronic and Computer Engineering University of Newcastle upon Tyne, 2006.  
[www.ukdesignforum.org/UKDF-Events/UKDF-2006/documents/AlexBystrov.pdf](http://www.ukdesignforum.org/UKDF-Events/UKDF-2006/documents/AlexBystrov.pdf), Acedido em Janeiro 2009.
- [CSD, 09] Gomes, Luís Filipe dos Santos, Acetatos teóricos da disciplina Conceção de Sistemas Digitais do Departamento de Engenharia Electrotécnica na Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa. <http://moodle.fct.unl.pt/>, Acedido em Janeiro 2009.
- [Cummings, 01] Clifford E. Cummings, “Synthesis and Scripting Techniques for Designing Multi-Asynchronous Clock Designs”, Synopsys Users Group (SNUG), Proceedings, 2001.  
[http://www.sunburst-design.com/papers/CummingsSNUG2001SJ\\_AsyncClk.pdf](http://www.sunburst-design.com/papers/CummingsSNUG2001SJ_AsyncClk.pdf), Acedido em Janeiro 2009.
- [EuroX10, 09] EuroX10, <http://www.eurox10.com>, Acedido em Janeiro 2009.
- [EXAME, 08] Isabel Infante, Reportagem Revista Exame Informática “Portugal faz bem software - Ambientes Inteligentes “ Agosto 2008.
- [FORDESIGN, 07] FORDESIGN – Formal Methods for Embedded Systems Co-Design. 2007. R&D Project POSC/EIA/61364/2004 com o apoio FCT - Fundação para a Ciência e a Tecnologia.  
<http://www.uninova.pt/fordesign/>, Acedido em Janeiro, 2009.
- [HAI, 09] Home Automation, Inc. <http://www.homeauto.com>, Acedido em Janeiro 2009.
- [IEEE, 06] Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs), 2006.  
<http://www.ieee802.org/15/pub/TG4.html>, Acedido em Janeiro 2009.
- [INSTEON, 05] INSTEON The Details, 2005,  
<http://www.insteon.net/pdf/insteonthedetails.pdf>, Acedido em Janeiro 2009.
- [KNX, 09] KNX Association, <http://www.knx.org>, Acedido em Janeiro 2009.



- [Manzano, 04] José Navarro Manzano, "Revisão e Discussão da Norma ISO 5807 - 1985" Faculdade Cantareira, São Paulo, 2004. <http://www.manzano.pro.br/alunado/navarro.pdf>, Acedido em Janeiro 2009.
- [Marranghello, 05] Marranghello, Norian, "Redes de Petri: Conceitos e Aplicações" Departamento de Ciencias De Computação e Estatística Universidade Estadual Paulista <http://www.dcce.ibilce.unesp.br/~norian/cursos/mds/ApostilaRdP-CA.pdf>, Acedido em Janeiro 2009.
- [Microchip, 09] Microchip Technology Inc. <http://www.microchip.com>, Acedido em Janeiro, 2009.
- [Moen, 03] Moen, Anders, "Introduction to Petri Nets", 2003, <http://www.ifi.uio.no/dbsem/2003vaar.html>, Acedido em Janeiro 2009
- [Noritake, 09] Noritake - KBC56A, <http://www.kbc56a.com>, Acedido em Janeiro 2009.
- [Nunes, 04a] Nunes, Renato; "Conceitos e Temas Associados aos Edifícios Inteligentes" 2004, <http://domobus.net/>, Acedido em Janeiro 2009.
- [Petri Nets World, 2009] <http://www.informatik.uni-hamburg.de/TGI/PetriNets/>, Acedido em Janeiro 2009.
- [RTP, 08] "Obra de Arte - Viver numa casa inteligente " Programa da RTP com a colaboração da DOMUS CONNECT e Ordem dos Engenheiros, Dezembro 2008.
- [SIC, 07] Reportagem SIC sobre "Domótica em Portugal", Maio 2007
- [SIC, 08] Reportagem SIC sobre "Domótica: casa inteligente ainda não é para todos", Março 2008
- [Stein, 03] Mike Stein, "Crossing the abyss: asynchronous signals in a synchronous world" EDN design feature, 2003. <http://www.edn.com/filtered/pdfs/contents/images/310388.pdf>, Acedido em Janeiro 2009.
- [TinyOS, 09] TinyOS. <http://tinyos.net/>, Acedido em Janeiro 2009.
- [Weinzierl, 09] Introduction to KNX. [http://www.weinzierl.de/download/Knx\\_Info.pdf](http://www.weinzierl.de/download/Knx_Info.pdf), Acedido em Janeiro 2009.
- [ZigBee, 06] ZigBee Specification, 2006. <http://www.zigbee.org/>, Acedido em Janeiro 2009

## Anexo I – Manual de utilização

Em seguida mostra-se o interface desenvolvido e como o utilizador pode configurar o sistema. O *browser* utilizado no desenvolvimento desta página foi Firefox 2. A experiência na navegação utilizando outros *browsers* pode variar, e até mesmo resultar na perda de certas funcionalidades como verificado com o uso do Internet Explorer. Inicialmente o utilizador abre o *browser* e indica o endereço do sistema. Este endereço pode ser o IP do coordenador que é apresentado no LCD da placa, ou se for dentro de uma rede local apenas apontar para <http://domsys/>, que por NetBIOS Name Service o endereço é descoberto (*resolved*).

Na primeira página surge uma descrição do sistema, mas sem a possibilidade de visionar ou alterar qualquer configuração.

Depois ao seleccionar alguma das outras secções da página aparecerá uma janela para validar o acesso (*login*), como mostrado na Figura I.1.

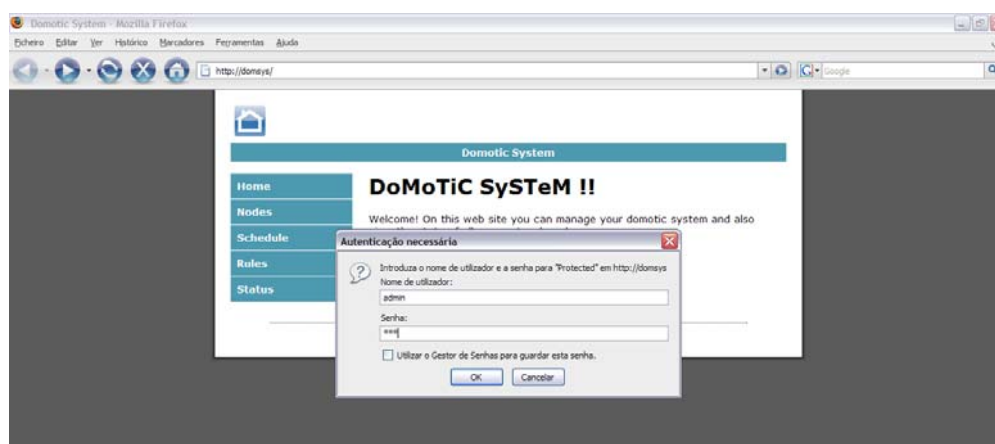


Figura I.0.1 – Acesso à configuração

Efectuando o *login* e acedendo à parte *Nodes* é possível ver os nós existentes nesta rede e/ou removê-los. É aqui que o instalador, após carregar no botão do nó para o registar, deverá atribuir o nome do nó de modo a saber no futuro que ID corresponde a cada nó. Este processo é apresentado na Figura I.2.

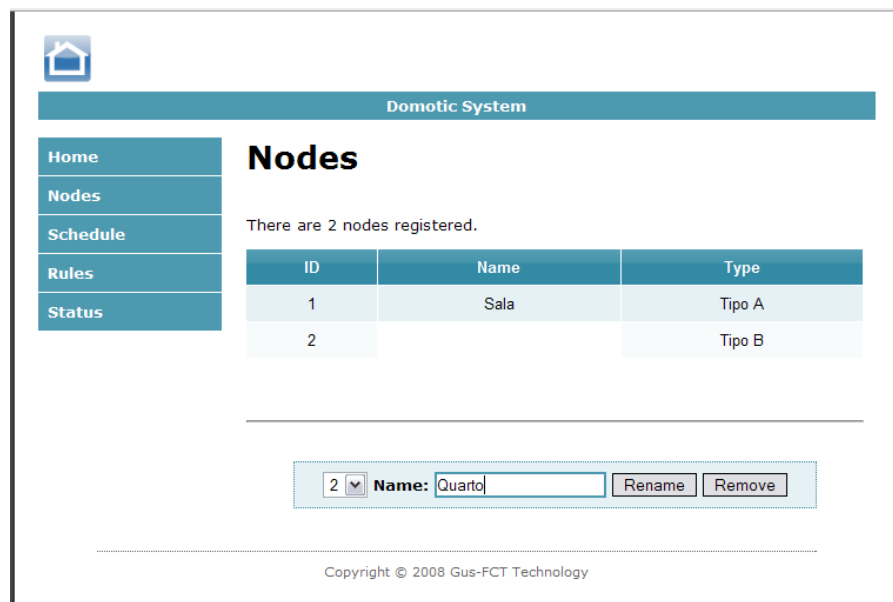



Figura I.0.2 – Secção dos nós

Para criar uma tarefa agendada o utilizador deve aceder a secção *Schedule* e aí adicionar a nova tarefa, como apresentado na Figura I.3. É também possível remover tarefas existentes. É apenas possível adicionar tarefas para nós já registados na rede e para datas futuras. Mostra-se ainda o tempo actual do sistema, pois ao aceder remotamente o utilizador pode não o ter presente. Conforme a escolha do dispositivo a accionar, as opções do valor vão sendo alteradas. O nome do dispositivo é na realidade um número que define o tipo, mas este é depois substituído por um texto existente no XML, utilizando AJAX. Estes automatismos garantem um interface dinâmico e apelativo ao utilizador. O utilizador deve ainda ao adicionar uma tarefa escolher a sua prioridade conforme a necessidade.



Domotic System

Home

Nodes

Schedule

Rules

Status

## Scheduled Tasks

The current system Time:

Date: Dec 04, 2008

Time: Thu 09:17:49

There are 1 action scheduled.

ID	Time	Type	Dest	Value
0	2008-12-25 10:30:30	TV	1	1

0 Remove

Time

HH:MM:SS 12 45 30

Date

YY:MM:DD 2008 12 26

Action

Destination 2 Device AC

Value 25°C Priority Normal

Add

Copyright © 2008 Gus-FCT Technology

Figura I.0.3 – Secção das tarefas agendadas

De modo a definir o conjunto de regras o utilizador deve aceder à secção *Rules*. Aí pode adicionar e remover regras, como se pode ver na Figura I.4. Mais uma vez à medida que certos campos vão sendo preenchidos, as opções vão sendo alteradas. O utilizador deve especificar o tipo de dispositivo e o nó a que a regra vai estar associado. Depois é necessário definir o valor limite e qual a condição a avaliar. É necessário ainda definir o tipo de regra a usar. É importante referir que quando se pretende utilizar o tipo de regra condicional os campos NodeID e Value têm outro sentido/utilidade. Neste caso específico estes definem quais os itens/regras que estão associadas à regra condicional.

Para toda a regra existe uma acção. A acção é definida pelo dispositivo e nó a actuar, assim como o valor e prioridade. As regras podem facultativamente ter uma “proacção”. Esta é uma acção que acontece depois da acção, quando a regra deixa de estar activa.

Item	Name	Type	ID	Device	Condition	Value	State	Action			
								Destination	Device	Value	Priority
0	LQuarto	Single	2	Presence	Not Active	0	0	2	Light	0	Normal

0 Remove

Rule

Name: AC-Sala
Type: Single Condition: Higher
NodeID: 1
Sensor: Temperature
Value: 30 °C

Action

Destination: 1
Device: AC
Value: 22°C
Priority: Normal

ProAction

Destination: 2
Device: AC
Value: Off
Priority: Normal

Add

Copyright © 2008 Gus-FCT Technology

Figura I.0.4 – Secção das regras

Para saber o estado dos sensores presentes na rede, o utilizador deve ir à secção *Status*. Aqui é mostrada uma lista de informação com todos os sensores e é ainda apresentado um gráfico com os últimos cinco valores recebidos para cada nó, como se pode ver na Figura I.5. Esta página actualiza-se automaticamente e tem como finalidade permitir ao utilizador acompanhar a evolução do sistema. A geração destes gráficos faz uso da tecnologia SVG.

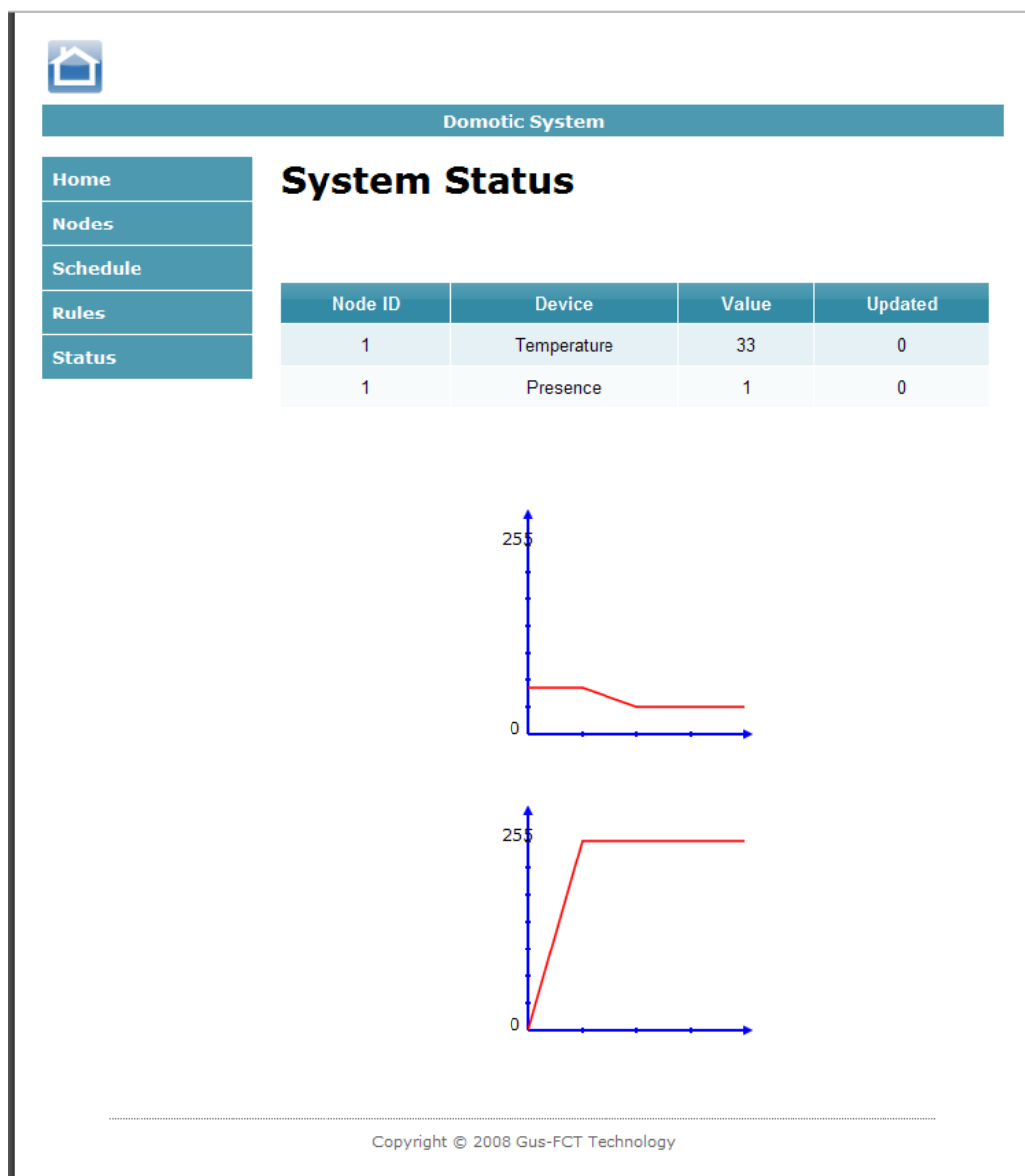


Figura I.0.5 – Secção do estado dos sensores